

89-4028



COMPUTER PROGRAMS
TO CALCULATE BASAL
AREA INCREMENT FROM
TREE RINGS

U.S. GEOLOGICAL SURVEY
Water-Resources Investigations Report 89-4028
1989

I 19.42/4:

READEX
FICHE

best copy available



COMPUTER PROGRAMS TO CALCULATE BASAL AREA INCREMENT FROM TREE RINGS

by Richard L. Phipps and Michael L. Field



U.S. GEOLOGICAL SURVEY
Water-Resources Investigations Report 89-4028
1989

DEPARTMENT OF THE INTERIOR

MANUEL LUJAN, JR., Secretary

U.S. GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

Chief Hydrologist
U.S. Geological Survey
Water Resources Division
461 National Center
Reston, Virginia 22092

Copies of this report can be
purchased from:

Western Distribution Branch
Open-File Services Section
U.S. Geological Survey
Box 25425, Federal Center
Lakewood, Colorado 80225

CONTENTS

	Page
Abstract.....	1
Introduction.....	1
Calculations performed by program AREA.....	2
Running the programs.....	4
Input files.....	5
Output files.....	7
Test run of tree-ring data from Signal Knob.....	8
References cited.....	8
Appendix A: Examples of AREA input and output files.....	13
Appendix B: Source code listings for programs.....	47

ILLUSTRATIONS

- Figure 1. Mean and 95-percent confidence limits of smoothed BAI (the nonclimatic component of BAI) of a pitch pine collection from Signal Knob near Front Royal, Virginia. Plot data are from the SIGNAL.CRN file created by AREA from the SIGNAL.RW and SIGNAL.RAD files. Data of the SIGNAL.RAD file are of incomplete radius lengths determined as the sum of the measured ring widths..... 9
2. Mean and 95-percent confidence limits of smoothed BAI of the Signal Knob collection. Plot data are from the KNOB.CRN file created by AREA from the SIGNAL.RW file and measurements of total radius contained in the KNOB.RAD file..... 9
3. Mean raw BAI of the Signal Knob collection. Based on ring width and total radius. Plot data are from the KNOB.CRN file..... 10
4. Mean tree-ring indices (climatic component) of the Signal Knob collection. Plot data are from the KNOB.CRN file..... 10

	Page
Figure 5. Ring widths of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.....	11
6. Raw BAI of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.....	11
7. Smoothed BAI of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.....	12
8. Tree-ring indices (climatic component) of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.....	12

COMPUTER PROGRAMS TO CALCULATE BASAL-AREA INCREMENT FROM TREE RINGS

by Richard L. Phipps and Michael L. Field

ABSTRACT

Two major programs, AREA and PLOT, and two minor programs, MAKERAD and TRING, intended for use on personal computers, are described. AREA performs preliminary data processing of ring-width data and calculates standardized ring widths and areas with and without the variance attributable to climate. PLOT provides a means of examining results from AREA graphically. MAKERAD calculates radius length from ring widths, and TRING arranges a string of ring-width data into a format acceptable by AREA. Examples are given of input to, and output from, program AREA.

INTRODUCTION

AREA is a computer program to describe growth quantity from tree rings. The program has been written to handle the preliminary data processing of tree-ring data on desktop computers. AREA calculates basal-area increment, BAI (Phipps and Whiton, 1988). AREA also calculates standardized ring widths, or indices (Fritts, 1976), and sets up files that may be used as inputs to other programs such as PLOT. PLOT is a program that provides the means for graphically examining raw ring widths, standardized indices, and BAI data for individual cores and for collections (mean chronologies). TRING takes ring-width data from measurement equipment or directly from the keyboard and arranges it into a standard format for input to AREA. MAKERAD calculates radius lengths from ring-width data and creates a file in a format that may be used as input to program AREA. The basic calculations and the input and output files for AREA are described. AREA, PLOT, MAKERAD and TRING are interactive and are not intended to require operating instructions.

The original version of AREA was written in FORTRAN¹ by Beth Cotter (presently with Electronic Data Systems, Herndon, Virginia). John C. Whiton (presently with the American Association of Dental Schools, Washington, D.C.) added some enhancements to AREA, including a routine to allow printer plots of results. The initial work of Cotter and Whiton provided the starting point from which the present programs have been developed. PLOT, MAKERAD, TRING, and the present version of AREA were all written by Michael Field.

¹ The use of brand names is for information, and does not constitute endorsement by the U.S. Geological Survey.

CALCULATIONS PERFORMED BY PROGRAM AREA

A raw ring-width series may be thought of as year-to-year variation in tree growth, correlative with climatic and other environmental factors, and superimposed on a growth trend. Thus, in a very generalized way, the year-to-year variation may be regarded as the climatic component, and the growth trend may be regarded as the nonclimatic component. An estimation of the nonclimatic component may be obtained by fitting a smooth curve to the raw ring-width series. This is commonly accomplished with a cubic spline smoothing function (Cook and Peters, 1981). The climatic component, w_c , can then be obtained by simply taking the difference between raw ring width, w_r , and the smoothed ring width, or nonclimatic component, w_s . A standardized climatic component, C , can be calculated by scaling the climatic component to the nonclimatic component:

$$w_c = w_r - w_s, \text{ and} \quad (1)$$

$$C = w_c / w_s$$

$$= (w_r / w_s) - 1. \quad (2)$$

Because year-to-year variation in ring width is roughly proportional to ring width, standardization has the effect of more-or-less evenly distributing variance with time.

The mean of a standardized climatic component series is 0 if the smoothing function has been perfectly applied. The minimum possible value of the standardized climatic component is -1.0 when $w_r = 0$. By adding 1.0 to the standardized climatic component, a standardized ring-width index, I_w , that contains no negative values, may be calculated:

$$I_w = C + 1$$

$$= w_r / w_s. \quad (3)$$

This is a form of the equation that has been used for years in dendrochronology to calculate indices. The indices may be thought of as an estimate of the standardized climatic component to which a constant of 1.0 has been added.

Because the indices are dimensionless, they are of little value in describing growth quantity. Though the nonclimatic component of ring width contains one dimension, this is also of little value unless a second dimension is added. For example, a wide ring represents more growth than a narrow ring, but only so long as both rings are on radii of approximately equal length. Tree-ring area is two-dimensional and, like basal area, BA (total cross sectional area at basal height), can be used to describe growth quantity. Just as ring width is an annual increment of radius length, tree-ring area can be referred to as basal-area increment, BAI.

Basal-area increment, BAI , is calculated from ring width, w , and radius length, R . BAI at year, t , is calculated from an equation of the difference in area between two circles:

$$BAI_t = \pi (R_t^2 - R_{t-1}^2) \quad (4)$$

where $R_{t-1} = R_t - w_t$. (5)

BAI is calculated in AREA inward from the outside year. Ring widths and radii are both handled in AREA as hundredths of millimeters, whereas BAI is handled as square centimeters.

Just as raw ring widths were described, raw basal area increment, BAI_r , may be thought of as composed of a climatic component superimposed on a non-climatic component. Again, as with ring widths, a cubic spline may be used to smooth BAI and produce an estimate of the nonclimatic component, BAI_s . Indices, or the climatic component, I_{BAI} , can then be calculated as the ratio of raw BAI to smoothed BAI :

$$I_{BAI} = BAI_r / BAI_s. \quad (6)$$

This is the method of calculating indices used in AREA. Indices calculated from BAI data are nearly the same, but not exactly equal to indices calculated from ring widths; that is, $I_w \neq I_{BAI}$. As already stated, the smoothed widths or areas are but estimates, or approximations, of the true nonclimatic component. One might be hard pressed to judge which of the two methods gives results that more accurately represent the true climatic component.

In practice, there are essentially no differences in indices of the mean collection chronologies calculated by the two methods. However, in AREA, calculations from BAI may give erroneously large index values for the innermost 6-10 years of some individual cores. If the slope of the smoothed BAI trend is great and the initial values of the trend are near zero, then calculation of indices involves division by numbers near zero. Division by numbers near zero seems to be the cause of the erroneously large index values. Until this problem is resolved, the initial 6-10 index values for cores in which initial BAI values are near zero should be disregarded.

The manner of fitting a smooth curve to either ring-width or raw BAI data is a bit of a guessing game. The objective to the guessing game is to fit a curve to the data such that indices calculated from the curve contain as much of the climatic component as possible with as little of the nonclimatic component as possible. Stating this in a different way, the more accurately the smooth curve represents the nonclimatic component, the more accurately the indices represent the climatic component. If the curve is too stiff, it might not include all of the nonclimatic component, meaning that the indices might contain considerably more than just climatic information. If the curve is too flexible, it might contain a significant portion of the climatic component, meaning that indices contain only a portion of the climatic information.

The climatic component (indices) can be correlated with climate. Intuitively, for any given tree-ring collection, the most accurate estimates of the nonclimatic component (smooth curves) are those that yield indices most highly

correlated with climate. Indeed, for the time being, though the nonclimatic component of either BAI or widths can be estimated without calculating indices, the accuracy of the estimates can only be tested by correlating indices with climatic data.

Program AREA uses a cubic spline to smooth BAI. The cubic spline may be described as removing 50 percent of the variance of the raw BAI for cycles less than a given number of years. Thus, any given spline may be described in terms of years. For example, a 20-year spline yields a more flexible curve than does a 200-year spline. Blasing and others (1981) have shown that for the typical deciduous forest collection in which most sampled trees are 150 to 250 years old, a 60-year spline applied to ring-width data seems to remove the most nonclimatic information while retaining the most climatic information in the indices. For this reason, a 60-year spline is customarily used with the BAI data. Though any spline may be specified in AREA, the one chosen is applied to data of all cores of the collection. As more is learned regarding the nonclimatic component, improved approximations of the nonclimatic component will no doubt be made, first by using a different spline for each core, and perhaps ultimately by using something other than a smoothing function. Because AREA can be run easily and quickly, certain insights may be gained into how certain trees are behaving by comparing results from a stiff spline (for example, 100-year), a more normal spline (60-year), and a flexible spline (20-year).

Smoothed BAI (the nonclimatic component) is individually determined for each core selected for calculation. The mean smoothed BAI is calculated by year as the mean of the smoothed BAI of all selected cores. The standard error of the mean of each year is calculated and adjusted by a multiplier from a *t*-table to give the 95-percent confidence limits of each yearly mean. This calculation is based on the number of cores, not the number of trees. If comparisons of data between cores within trees indicate less independence than data between trees, then a more accurate estimate of the 95-percent confidence limits could be obtained by basing the calculations on number of trees instead of number of cores.

BAI calculations are based on the assumption that the transverse area of a tree ring can be estimated as the difference between two perfect circles where ring width is the difference in radius length between the two circles (eq. 4). A better estimate may be obtained by using more than one ring width. However, the improved estimate is more accurate if the areas are individually calculated for each width and then averaged, rather than averaging the widths before calculating area.

RUNNING THE PROGRAMS

The programs may be placed on a single 360K floppy disk. Two minor programs that can also be included on the same disk are TRING and MAKERAD. Compiled versions of the programs are compact enough to allow inclusion on the program disk of a menu program enabling any of the programs to be accessed from a single menu. Source code listings of all four programs, AREA, PLOT, MAKERAD, and TRING, are included in Appendix B.

The programs, AREA and PLOT, were written in C programming language, compiled using Microsoft C 5.0, and have been run with both DOS and MS-DOS. However, a graphics card is needed for PLOT. The screen plots are only in black and white with a CGA card, but are in selectable colors with an EGA card. PLOT was written for, and has only been run on, the HP-7475A plotter. PLOT should, however, work on a number of other HP plotter models and possibly on some other brands that use the HP-GL instruction set.

Input Files

Just as the area of a circle is a function of radius length, the area of a tree ring, BAI, is dependent upon radius length. Thus, in addition to a ring-width file, AREA requires radius data. Radius data may be input to AREA from a file created by either AREA or MAKERAD, or may be keyed in directly while executing AREA.

1. [NAME].RAD file. The radius file contains data of total radius length of each core to be used in calculations. A radius file may be created by either AREA or MAKERAD. These programs calculate radius length as the sum of ring widths of each core. Thus, this method of calculating radii is accurate only if the cores are measured to pith. If the cores from which the width data were obtained were not measured to pith, or there are other reasons to use radius lengths not equal to the sum of the ring widths, radius lengths may be entered into AREA by hand (no radius file is needed or created), or entered into AREA as a special file. A radius file may be set up as follows:

- a. File name -- 1-8 alphanumeric characters with a .RAD extension.
- b. First line of file -- File title or message.
- c. Data format -- The first 8 columns (left justified) are the core ID number. The next 6 columns (right justified) are the radius length in hundredths of millimeters. Enter data for one core per line. The following data are for two pairs of cores from a .RAD file created by MAKERAD of a white oak collection obtained at Limberlost in the Shenandoah National Park:

LIMBERLOST WHITE OAK

10031	25553	361	1618	1978	1779
10032	31729	362	1617	1978	1732
10051	26234	392	1587	1978	1696
10052	27986	385	1594	1978	1690

In addition to core ID (identification number) and radius length, a .RAD file created by MAKERAD will include (unlabeled) number of rings, inside date, outside date, and year at which a radius length of 10 cm is reached. This last value is estimated to be the radius length at which many deciduous species reach canopy size in a mixed-age stand.

A .RAD file created directly by AREA will contain the same information as one created by MAKERAD with the exception that the 10-cm radius is only a default value. That is, any length, including zero, may be specified.

The Limberlost cores, used in the example above, were not measured to pith. Total radius of each core was measured with a millimeter rule, and the data were entered into a special .RAD file. Lines from the special file, corresponding to the example above, were:

LIMBERLOST WHITE OAK

10031	27200
10032	34000
10051	32000
10052	38000

Radius is given in mm x 10^{-2} . Hence, the radius of core number 10031, measured to the nearest mm as 272, was entered into the special file as 27200.

AREA will only calculate data for cores specified in the radius file. Thus, even if a ring-width file is quite large, it is possible to calculate data for only a few selected cores by setting up a special radius file containing only radii of the selected cores. For example, MAKERAD may be used to calculate radii of all the cores in the ring-width file of a collection. A word processor can then be used to edit the file to take out the radii of the cores in which there is no interest. Core ID's used in a radius file should be arranged in the same sequential order as the cores in the ring-width file.

2. [NAME].RW file. The ring-width file is composed of measurement data usually generated from some type of tree-ring measurement equipment. Program TRING (included in Appendix B) takes ring-width measurements from either keyboard entry or measurement equipment and arranges the data into the following format:

a. File name -- 1-8 alphanumeric characters with a .RW extension.

b. Data format -- similar to the standard Arizona format for ring-width data. The Laboratory of Tree-Ring Research at the University of Arizona has established a format for input of ring-width data to all of their programs for standardizing tree rings.

The first line of the file is a file title or message. CAUTION: Arizona files may not include a file title line.

The rest of the file is composed of data. Originally, Arizona's data were input on punch cards for use in FORTRAN programs. Thus, the format is column dependent. The following example was modified from a ring-width file of pitch pine cores collected at Signal Knob near Front Royal, Virginia:

SIGNAL KNOB PITCH PINE

SIG022	1843	142	218	187	219	290	306	188				
SIG022	1850	192	207	280	300	229	178	156	208	225	141	
SIG022	1860	52	125	152	99900							
SIG131	1968	27	29									
SIG131	1970	32	24	26	31	36	15	5	23	19	16	
SIG131	1980	0	0	99900								

The first 8 columns of each line (left justified) contain the core ID number. The rest of the line contains data for a single decade. Columns 8-12 are the

beginning year of data for the decade. In the example, the first year of data of the 1840's decade for sample SIG022 is 1843. Ring-width data (in mm X 10²) are entered (right justified) in 10 fields of 6. If the year of the last ring width of a core is not a year ending in 9, then the following year is entered as 99900. Some Arizona-type files may use 999 instead of 99900.

In summary, if ring-width files created by an Arizona program are being used, the files should be checked to be sure that they contain:

- 1) a title line, and
- 2) 99900 as end of core code.

3. [NAME].ELW file. AREA can handle ring-width data that have been separated into two measurements per ring. The format for the data file is similar to that described above for total ring width except that a full line contains data for a pentad, not a decade. On each line, the odd entries (1st, 3rd, and so forth) are for the inner portion of each ring (earlywood) and the even entries (2nd, 4th, and so forth) are for the outer portion (latewood). A few lines of data from a file of measurements of a loblolly pine collection from the Great Dismal Swamp on the Virginia - North Carolina border are given as an example:

Lynn Ditch loblolly - earlywood/latewood

		earlywood/latewood											
3562	1828	205	31	549	75								
3562	1830	200	140	422	134	242	67	269	130	302	123		
3562	1835	273	150	180	111	83	94	139	80	147	122		
3562	1965	27	22	25	27	41	19	24	15	8	18		
3562	1970	14	6	5	10	14	6	7	9	99900			

The outside year of the example is 1973.

Output Files

Program AREA always creates a mean collection chronology file (.CRN) in our own format and a mean chronology file of indices (.IX1) in the Arizona format.

1. [NAME].CRN file. Mean collection chronology file that contains number of cores by year, raw BAI, smooth BAI, upper and lower confidence limits of the yearly means of the smoothed data, and ring-width indices. Means are by year for the range of years specified while executing AREA. This file may be printed in two different formats from program PLOT.
2. [NAME].IX1 file. Index file #1, composed of mean chronology indices for the range of years specified. This file is in the standard Arizona format for indices, and is intended for use as an input file for other programs (such as RESPON).
3. [NAME].IND file. Program AREA will, as options, create files containing data of individual cores. This file contains original ring widths, raw

BAI, smoothed BAI, and indices of all individual cores specified in the radius file. This file includes data of all years of each core regardless of what years were specified for the .CRN file. Because no means are calculated, no confidence limits are included. If the radius file contains data of many cores, and the ring-width data for these cores represent many years, this file can be quite long.

4. [NAME].IX2 file. Index file #2 contains indices of all individual cores in the standard Arizona format suitable for use as input to other programs (such as COFECHA).

TEST RUN OF TREE-RING DATA FROM SIGNAL KNOB

Tree-ring increment cores were taken from pitch pine (*Pinus rigida* Mill.) trees at Signal Knob at the northern end of Massanutten Mountain near Front Royal, Virginia. The samples were collected by John Whiton prior to the growing season in 1982. Two cores were collected from each of 16 trees. The cores were sanded, crossdated, and measured in accordance with standard procedures (Phipps, 1985; Stokes and Smiley, 1968). At least one core from each of four of the trees was unacceptable for measurement. The final collection is composed of 2 cores from each of 12 trees.

Signal Knob data were used as input data files to AREA and the output files from AREA were used in PLOT to create graphs of the results. Both the input and output files for the Signal Knob collection are contained in Appendix A. Examples of graphs produced by PLOT of Signal Knob data calculated by AREA are presented in figures 1-8.

REFERENCES CITED

- Blasing, T. J. and Duvick, D. N., 1983, Filtering the effects of competition from ring-width series: *Tree-Ring Bulletin*, v. 43, p. 19-30.
- Cook, E. R. and Peters, K., 1981, The smoothing spline: a new approach to standardizing forest interior tree-ring width series for dendroclimatic studies. *Tree-Ring Bulletin*: v. 41, p. 45-53.
- Fritts, H. C., 1976, *Tree Rings and climate*: Academic Press, London, 567p.
- Phipps, R. L., 1985, Collecting, preparing, crossdating, and measuring tree increment cores. U.S. Geological Survey Water Resources Investigations Report 85-4148, Reston, VA., 48p.
- Phipps, R. L. and Whiton, J. C., 1988, Decline in long-term growth trends of white oak: *Canadian Journal of Forest Research*, v. 18, p. 24-32.
- Stokes, M. A. and Smiley, T. L., 1968, *An introduction to tree-ring dating*. University of Chicago Press, Chicago, 73p.

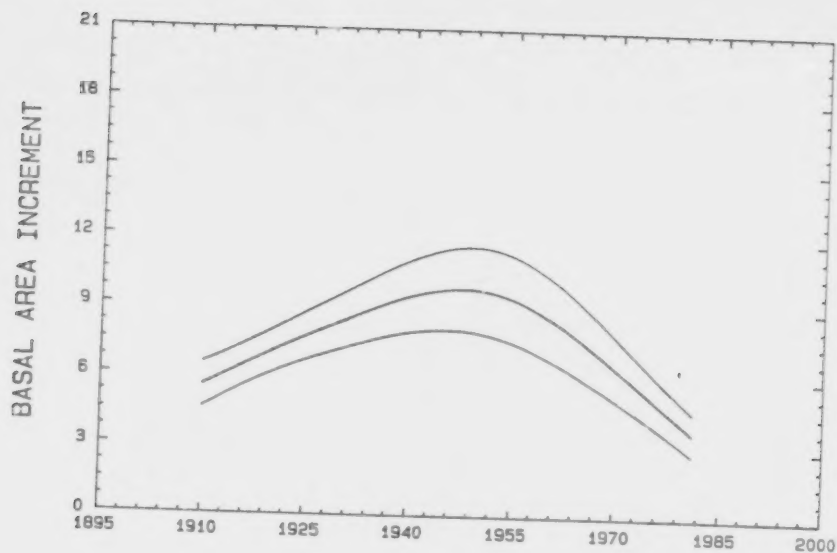


Figure 1. Mean and 95-percent confidence limits of smoothed *BAI* (the nonclimatic component of *BAI*) of a pitch pine collection from Signal Knob near Front Royal, Virginia. Plot data are from the SIGNAL.CRN file created by AREA from the SIGNAL.RW and SIGNAL.RAD files. Data of the SIGNAL.RAD file are of incomplete radius lengths determined as the sum of the measured ring widths.

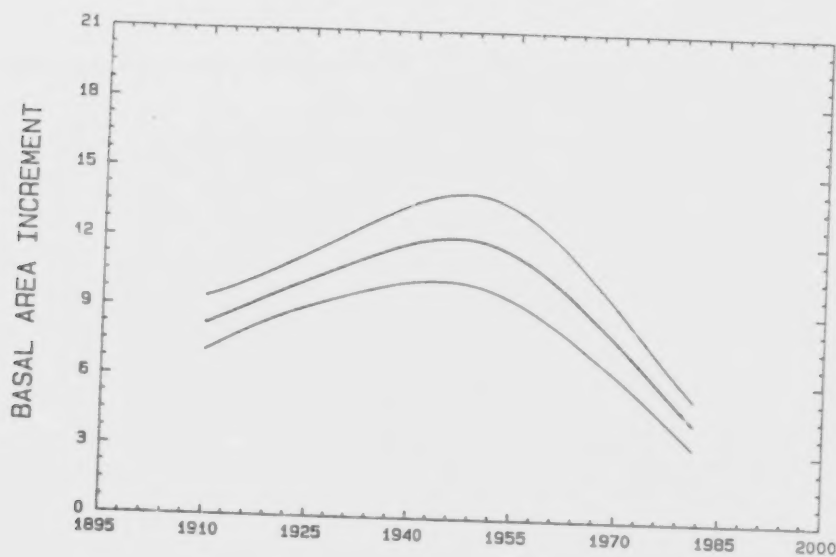


Figure 2. Mean and 95-percent confidence limits of smoothed *BAI* of the Signal Knob collection. Plot data are from the KNOB.CRN file created by AREA from the SIGNAL.RW file and measurements of total radius contained in the KNOB.RAD file.

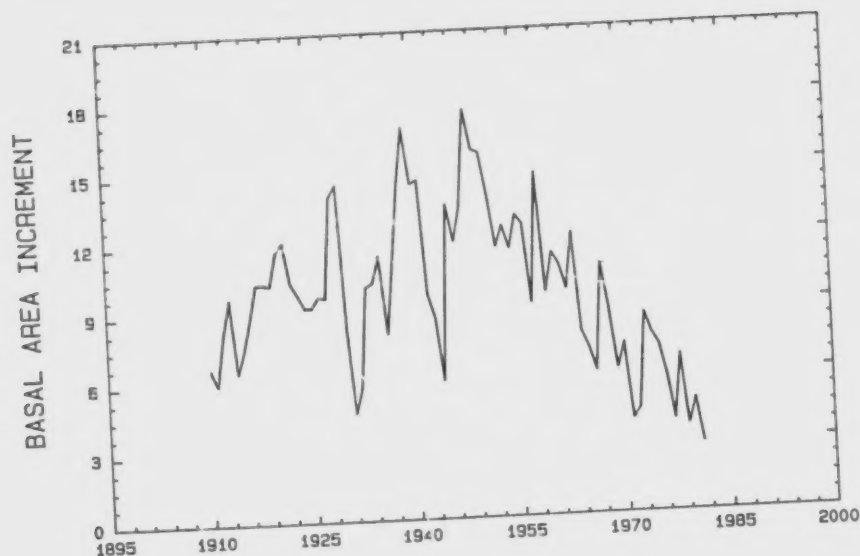


Figure 3. Mean raw BAI of the Signal Knob collection. Based on ring width and total radius. Plot data are from the KNOB.CRN file.

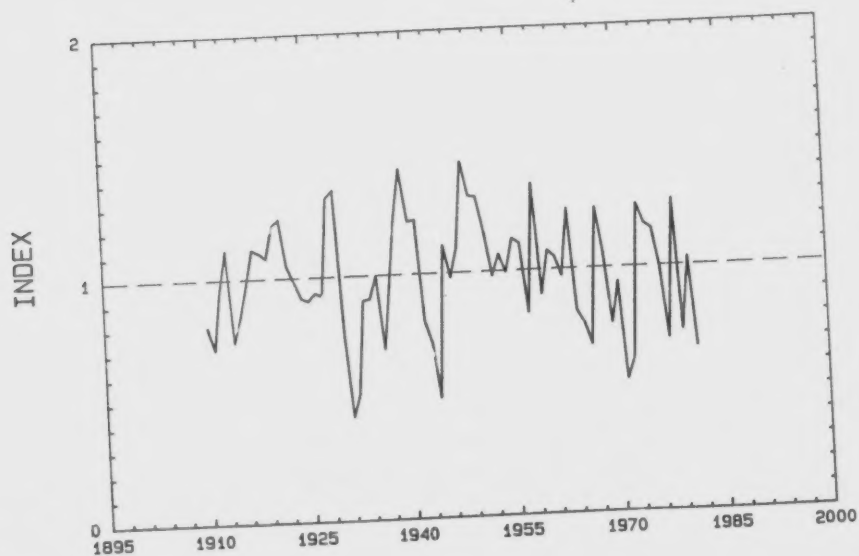


Figure 4. Mean tree-ring indices (climatic component) of the Signal Knob collection. Plot data are from the KNOB.CRN file.

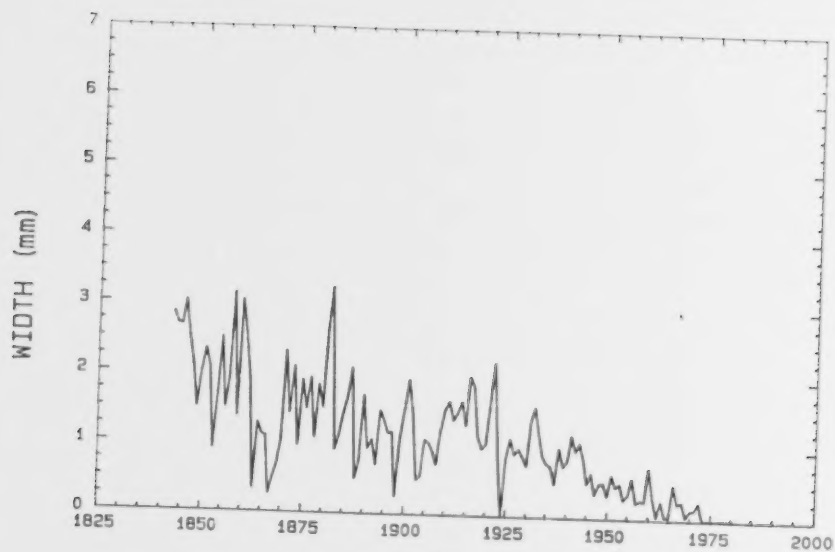


Figure 5. Ring widths of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.

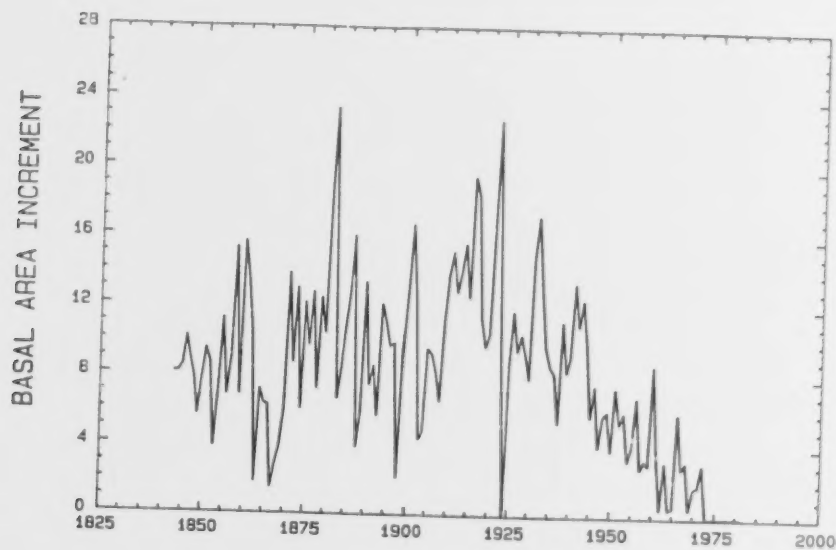


Figure 6. Raw BAI of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.

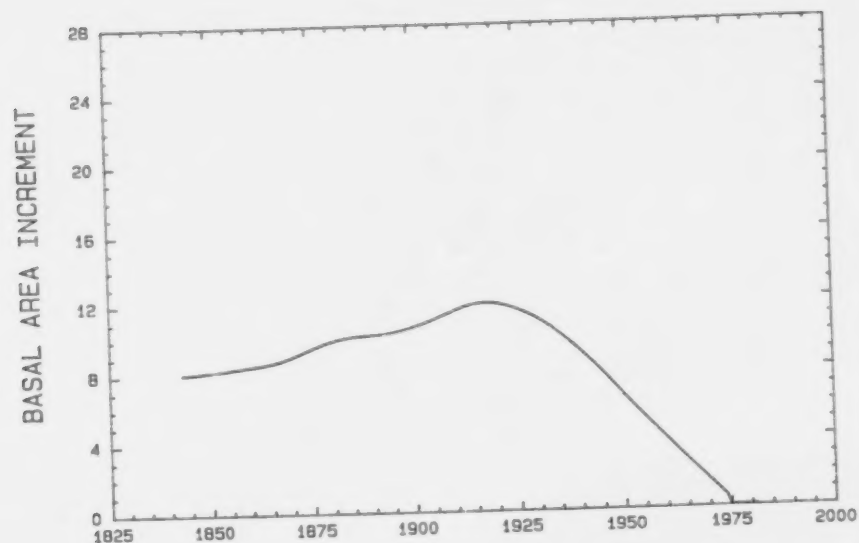


Figure 7. Smoothed BAI of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.

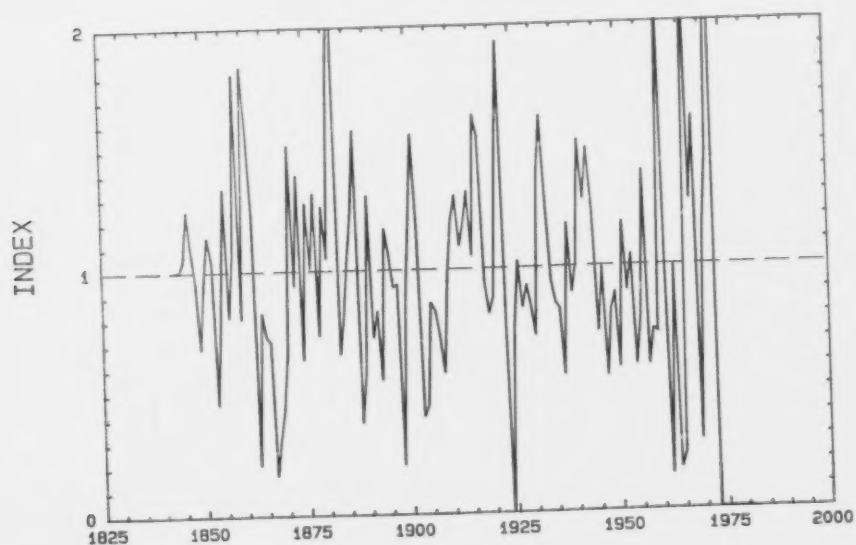


Figure 8. Tree-ring indices (climatic component) of sample SIG 02-1 of the Signal Knob collection. Plot data are from the KNOB.IND file.

APPENDIX A EXAMPLES OF AREA INPUT AND OUTPUT FILES

Input files:	Page
SIGNAL.RW.....	15
SIGNAL.RAD.....	19
KNOB.RAD.....	19
Output files:	
SIGNAL.CRN.....	20
KNOB.CRN.....	21
KNOB.IND.....	23
KNOB.IX1.....	42
KNOB.IX2.....	42

Input file SIGNAL.RW

SIGNAL	KNOB	PITCH	PINE																	
SIG01-1	1900	119	177	133	349	235	38	93	210	274	284									
SIG01-1	1910	140	109	204	310	183	94	162	160	191	293									
SIG01-1	1920	244	231	178	198	163	209	207	244	312	312									
SIG01-1	1930	131	107	111	215	168	204	179	248	322	346									
SIG01-1	1940	240	313	185	187	107	317	285	288	324	263									
SIG01-1	1950	247	258	213	180	197	255	229	176	214	178									
SIG01-1	1960	232	212	144	181	141	105	119	255	172	116									
SIG01-1	1970	116	59	44	139	106	77	80	58	102	74									
SIG01-1	1980	88	42	99900																
SIG01-2	1890	115	133	176	133	145	106	134	162	263	155									
SIG01-2	1900	98	128	89	156	177	13	43	100	166	167									
SIG01-2	1910	78	48	95	191	127	54	70	99	93	158									
SIG01-2	1920	149	142	140	174	161	193	176	164	227	203									
SIG01-2	1930	106	51	66	141	157	241	199	259	221	311									
SIG01-2	1940	259	240	122	172	102	287	247	243	345	221									
SIG01-2	1950	177	173	171	186	206	202	156	117	202	183									
SIG01-2	1960	253	210	107	145	110	103	86	220	170	122									
SIG01-2	1970	101	48	62	149	123	118	69	44	96	54									
SIG01-2	1980	80	43	99900																
SIG02-1	1850	284	269	267	302	251	212	148	199	233	206									
SIG02-1	1860	89	161	250	148	190	314	136	304	254	191									
SIG02-1	1870	32	128	112	108	25	49	67	102	231	142									
SIG02-1	1880	209	95	190	149	193	107	183	151	259	324									
SIG02-1	1890	90	112	143	166	208	49	75	169	93	106									
SIG02-1	1900	70	148	134	116	117	25	111	150	192	141									
SIG02-1	1910	49	55	107	103	91	72	121	150	162	136									
SIG02-1	1920	147	162	129	198	185	112	96	103	165	219									
SIG02-1	1930	82	0	83	111	89	97	86	73	137	157									
SIG02-1	1940	119	87	77	73	48	99	73	81	117	95									
SIG02-1	1950	107	85	49	64	34	49	51	32	62	45									
SIG02-1	1960	50	27	34	57	23	27	25	72	30	4									
SIG02-1	1970	26	4	5	49	23	26	4	14	15	25									
SIG02-1	1980	0	0	99900																
SIG02-2	1843	142	218	187	219	290	306	188												
SIG02-2	1850	192	207	280	300	229	178	156	208	225	141									
SIG02-2	1860	52	125	152	108	156	269	100	203	141	156									
SIG02-2	1870	33	100	60	80	21	25	44	103	186	174									
SIG02-2	1880	171	94	122	133	128	109	136	112	148	236									
SIG02-2	1890	72	119	88	112	124	39	52	72	45	41									
SIG02-2	1900	35	75	94	46	62	13	56	84	95	80									
SIG02-2	1910	36	40	63	42	56	48	83	116	117	98									
SIG02-2	1920	84	113	78	57	60	49	46	52	68	103									
SIG02-2	1930	63	0	45	48	51	87	51	56	96	103									
SIG02-2	1940	91	80	63	59	39	71	59	72	102	62									
SIG02-2	1950	97	73	49	55	45	51	61	57	76	55									
SIG02-2	1960	75	53	49	57	20	14	14	27	39	17									
SIG02-2	1970	24	14	18	55	25	34	6	18	17	24									
SIG02-2	1980	0	0	99900																
SIG03-1	1890	388	175	147	216	211	310	71	108	248	320									
SIG03-1	1900	180	48	180	171	232	228	92	171	208	203									
SIG03-1	1910	40	52	156	228	124	211	235	243	221	201									
SIG03-1	1920	235	197	184	217	192	187	190	168	312	286									
SIG03-1	1930	110	35	60	191	232	217	132	189	281	357									
SIG03-1	1940	263	273	111	124	45	155	180	180	250	200									
SIG03-1	1950	92	175	90	188	143	115	172	118	137	67									
SIG03-1	1960	84	84	125	127	70	89	88	101	54	43									
SIG03-1	1970	69	30	29	112	54	117	84	50	61	12									
SIG03-1	1980	71	29	99900																
SIG03-2	1880	521	445	377	266	117	221	332	179	172	256									
SIG03-2	1890	119	115	179	183	273	88	67	213	154	142									
SIG03-2	1900	44	110	87	173	195	20	101	152	184	181									
SIG03-2	1910	43	35	81	111	98	141	206	191	191	208									
SIG03-2	1920	188	187	143	140	140	123	123	102	238	218									
SIG03-2	1930	100	21	55	121	137	124	61	98	177	293									
SIG03-2	1940	222	272	121	135	52	131	146	98	162	146									
SIG03-2	1950	131	114	71	120	148	88	135	97	124	34									
SIG03-2	1960	55	57	66	91	28	32	32	53	56	44									
SIG03-2	1970	62	40	33	74	47	56	62	31	33	9									
SIG03-2	1980	65	15	99900																

Input file SIGNAL.RW, continued

SIG05-1 1910	325	362	329	512	319	284	346	411	379	353
SIG05-1 1920	529	527	499	380	309	319	278	379	438	385
SIG05-1 1930	160	145	126	243	242	323	249	296	333	370
SIG05-1 1940	330	303	170	106	106	237	274	328	395	681
SIG05-1 1950	738	376	292	262	216	287	248	214	316	182
SIG05-1 1960	191	209	249	360	209	165	126	256	213	145
SIG05-1 1970	117	95	67	110	104	68	85	47	99	40
SIG05-1 1980	34	19	99900							
SIG05-2 1880	284	350	205	139	223	243	291	210	264	349
SIG05-2 1890	48	19	110	167	178	99	116	191	208	212
SIG05-2 1900	29	69	69	127	157	15	51	137	128	176
SIG05-2 1910	114	96	114	116	110	120	171	256	161	211
SIG05-2 1920	206	199	157	203	161	236	209	219	302	259
SIG05-2 1930	180	206	182	285	222	208	138	193	241	218
SIG05-2 1940	240	266	259	153	136	167	182	180	286	166
SIG05-2 1950	184	206	173	168	173	227	209	118	183	110
SIG05-2 1960	177	158	145	137	110	126	96	144	108	68
SIG05-2 1970	65	28	43	84	100	71	47	31	88	19
SIG05-2 1980	17	6	99900							
SIG06-1 1882	483	268	210	172	313	237	230	360		
SIG06-1 1890	52	59	107	147	196	119	195	263	218	193
SIG06-1 1900	114	182	158	225	159	109	204	234	293	160
SIG06-1 1910	136	141	155	213	117	167	183	218	178	158
SIG06-1 1920	141	126	133	190	163	165	158	116	184	185
SIG06-1 1930	96	0	72	126	147	159	81	124	156	201
SIG06-1 1940	171	135	85	70	54	103	116	140	147	127
SIG06-1 1950	138	112	135	102	98	95	92	61	107	48
SIG06-1 1960	82	62	65	91	51	46	47	66	31	38
SIG06-1 1970	62	41	33	100	66	63	32	38	53	20
SIG06-1 1980	27	22	99900							
SIG06-2 1880	289	300	341	196	189	213	627	279	204	234
SIG06-2 1890	17	55	102	110	207	134	153	224	196	158
SIG06-2 1900	92	157	126	255	178	120	161	191	220	156
SIG06-2 1910	106	105	138	155	82	129	141	168	132	145
SIG06-2 1920	114	120	107	124	138	149	111	86	158	152
SIG06-2 1930	81	0	60	86	121	131	89	119	148	187
SIG06-2 1940	137	115	45	76	46	104	133	129	122	98
SIG06-2 1950	132	102	97	85	92	86	95	73	106	64
SIG06-2 1960	100	87	71	65	44	45	45	61	59	48
SIG06-2 1970	47	45	44	80	55	61	52	30	78	36
SIG06-2 1980	52	32	99900							
SIG08-1 1887	471	358	661							
SIG08-1 1890	126	148	200	228	235	136	194	270	255	180
SIG08-1 1900	146	183	181	231	195	158	159	248	278	183
SIG08-1 1910	183	171	171	199	151	142	198	157	193	171
SIG08-1 1920	210	207	171	129	122	84	131	114	180	220
SIG08-1 1930	134	172	125	174	166	125	127	165	151	220
SIG08-1 1940	207	247	116	114	88	249	140	141	237	207
SIG08-1 1950	250	184	166	196	118	172	158	91	208	148
SIG08-1 1960	139	166	111	161	118	115	96	161	135	63
SIG08-1 1970	108	38	88	128	133	47	75	69	99	30
SIG08-1 1980	68	41	99900							
SIG08-2 1890	58	97	179	200	205	122	185	262	258	225
SIG08-2 1900	149	200	169	170	220	147	229	177	186	184
SIG08-2 1910	139	107	142	125	101	139	173	149	207	199
SIG08-2 1920	196	204	154	120	94	99	139	130	196	224
SIG08-2 1930	158	119	121	173	154	148	106	184	184	251
SIG08-2 1940	153	162	113	109	69	196	211	194	259	195
SIG08-2 1950	190	138	177	179	126	181	174	96	140	86
SIG08-2 1960	117	136	90	132	120	100	93	140	119	53
SIG08-2 1970	99	41	79	116	156	53	80	57	59	20
SIG08-2 1980	56	18	99900							
SIG09-1 1890	179	181	187	251	256	97	88	116	192	174
SIG09-1 1900	127	182	122	201	140	62	150	200	228	137
SIG09-1 1910	182	159	191	233	96	171	156	236	183	158
SIG09-1 1920	224	172	110	96	147	146	127	130	190	175
SIG09-1 1930	84	0	16	82	82	115	112	167	198	188
SIG09-1 1940	183	185	102	120	42	152	142	124	144	106
SIG09-1 1950	110	124	84	114	93	126	86	70	106	75
SIG09-1 1960	93	78	85	90	46	48	43	111	99	63

Input file SIGNAL.RW, continued

SIG09-1 1970	67	0	48	38	25	48	32	20	40	16
SIG09-1 1980	34	30	99900							
SIG09-2 1890	341	287	310	259	255	238	82	135	127	199
SIG09-2 1900	154	127	206	152	207	194	118	160	246	208
SIG09-2 1910	247	243	230	248	246	118	175	171	223	183
SIG09-2 1920	246	249	286	198	205	191	183	226	146	237
SIG09-2 1930	232	139	68	111	124	177	165	203	213	243
SIG09-2 1940	213	242	228	127	99	255	176	266	293	214
SIG09-2 1950	211	190	144	149	148	163	130	98	178	94
SIG09-2 1960	135	116	120	110	68	60	68	110	144	120
SIG09-2 1970	105	77	14	90	72	103	76	41	59	62
SIG09-2 1980	82	64	99900							
SIG11-1 1863	65	105	84	156	166	207	224			
SIG11-1 1870	205	15	60	91	88	138	222	220	323	231
SIG11-1 1880	314	239	413	250	289	190	276	200	234	258
SIG11-1 1890	214	204	193	210	224	141	136	183	191	133
SIG11-1 1900	98	114	93	195	144	144	155	173	184	225
SIG11-1 1910	146	124	156	170	106	136	142	179	128	128
SIG11-1 1920	142	167	149	78	82	90	109	106	180	158
SIG11-1 1930	86	0	31	80	48	84	43	82	130	155
SIG11-1 1940	142	148	100	109	76	164	109	151	179	186
SIG11-1 1950	160	163	133	117	133	133	136	102	174	105
SIG11-1 1960	110	94	100	108	63	67	31	100	60	50
SIG11-1 1970	56	26	48	70	96	111	109	28	54	42
SIG11-1 1980	25	0	99900							
SIG11-2 1870	7	40	58	144	115	159	193	234	386	261
SIG11-2 1880	246	219	375	216	214	133	263	240	262	244
SIG11-2 1890	157	122	146	190	213	124	76	174	149	112
SIG11-2 1900	41	62	80	148	138	97	95	192	251	200
SIG11-2 1910	107	120	136	158	64	107	113	114	99	110
SIG11-2 1920	107	83	68	83	70	55	81	79	108	104
SIG11-2 1930	52	0	24	47	52	71	37	51	65	110
SIG11-2 1940	86	113	89	67	42	106	66	78	120	130
SIG11-2 1950	115	113	80	105	73	80	103	69	104	73
SIG11-2 1960	49	67	52	83	49	62	34	65	41	29
SIG11-2 1970	63	14	26	62	66	75	64	40	39	46
SIG11-2 1980	45	16	99900							
SIG12-1 1882	302	314	300	308	206	301	323	322		
SIG12-1 1890	61	68	121	141	185	38	37	87	189	154
SIG12-1 1900	28	130	116	194	194	288	274	201	230	209
SIG12-1 1910	258	207	352	227	167	248	141	213	230	191
SIG12-1 1920	264	211	177	46	50	77	126	106	171	174
SIG12-1 1930	93	21	51	72	132	143	144	163	180	187
SIG12-1 1940	137	117	124	88	53	76	40	49	166	152
SIG12-1 1950	127	158	120	117	122	113	118	80	178	145
SIG12-1 1960	124	127	73	174	77	47	56	84	105	93
SIG12-1 1970	80	30	39	74	107	127	79	74	114	49
SIG12-1 1980	125	116	99900							
SIG12-2 1900	28	91	50	108	99	179	226	232	311	227
SIG12-2 1910	251	188	261	269	150	210	143	206	180	167
SIG12-2 1920	189	166	161	25	38	65	130	105	156	152
SIG12-2 1930	58	22	52	99	144	125	106	143	132	150
SIG12-2 1940	112	106	136	83	45	110	107	130	156	141
SIG12-2 1950	123	103	134	135	129	130	87	62	141	82
SIG12-2 1960	112	96	82	125	60	55	53	81	94	67
SIG12-2 1970	74	44	59	85	80	77	35	35	80	20
SIG12-2 1980	46	55	99900							
SIG13-1 1870	24	70	138	192	150	188	257	347	459	354
SIG13-1 1880	393	326	519	333	391	351	494	388	402	346
SIG13-1 1890	212	227	252	295	289	186	153	224	201	164
SIG13-1 1900	94	139	73	99	91	95	88	140	150	123
SIG13-1 1910	133	113	142	153	82	109	135	143	119	105
SIG13-1 1920	147	172	176	156	135	139	135	116	153	148
SIG13-1 1930	89	59	77	125	116	127	93	140	211	184
SIG13-1 1940	158	165	103	103	72	151	115	145	160	127
SIG13-1 1950	165	129	108	117	122	108	101	88	182	113
SIG13-1 1960	83	64	55	63	49	35	30	56	27	29
SIG13-1 1970	32	29	23	39	31	15	19	19	20	17
SIG13-1 1980	9	0	99900							
SIG13-2 1870	35	64	111	208	208	198	242	290	467	404

Input file SIGNAL.RW, continued

SIG13-2 1880	384	359	570	279	331	242	435	367	446	428
SIG13-2 1890	258	198	217	270	330	173	197	162	149	103
SIG13-2 1900	42	49	62	111	93	89	92	144	170	131
SIG13-2 1910	96	70	113	121	69	83	102	80	64	96
SIG13-2 1920	122	138	111	129	109	90	102	87	133	142
SIG13-2 1930	87	66	51	86	91	111	57	90	126	120
SIG13-2 1940	108	73	52	44	39	60	46	71	87	84
SIG13-2 1950	88	102	85	78	62	102	133	101	119	76
SIG13-2 1960	62	76	96	74	52	34	36	46	43	30
SIG13-2 1970	32	24	26	31	36	15	5	23	19	16
SIG13-2 1980	0	0	99900							
SIG14-1 1880	560	379	312	295	179	284	390	340	311	346
SIG14-1 1890	192	103	159	183	215	202	214	352	285	271
SIG14-1 1900	46	84	86	159	192	172	207	248	235	213
SIG14-1 1910	143	129	148	196	134	154	155	172	223	265
SIG14-1 1920	216	201	158	141	121	161	148	170	260	229
SIG14-1 1930	134	150	110	166	196	171	83	129	174	228
SIG14-1 1940	176	161	79	69	73	120	135	147	200	162
SIG14-1 1950	172	142	125	123	154	142	111	79	107	68
SIG14-1 1960	80	73	88	104	74	75	60	67	70	40
SIG14-1 1970	57	35	57	67	67	73	67	48	63	48
SIG14-1 1980	41	18	99900							
SIG14-2 1880	507	378	321	296	269	351	897	441	295	436
SIG14-2 1890	190	151	160	271	289	219	252	313	289	289
SIG14-2 1900	24	45	86	107	158	145	220	223	295	225
SIG14-2 1910	116	76	110	189	152	158	201	229	210	182
SIG14-2 1920	279	349	224	189	171	159	182	182	295	276
SIG14-2 1930	174	126	96	213	207	191	69	104	183	196
SIG14-2 1940	209	179	117	93	78	157	133	195	224	221
SIG14-2 1950	161	164	137	139	142	128	113	93	135	92
SIG14-2 1960	109	110	93	127	102	82	61	74	43	20
SIG14-2 1970	37	21	34	62	53	37	44	31	44	30
SIG14-2 1980	24	8	99900							
SIG16-1 1862	265	424	241	212	159	205	210	198		
SIG16-1 1870	25	68	161	197	45	66	160	133	250	191
SIG16-1 1880	294	194	281	244	229	172	266	220	177	215
SIG16-1 1890	49	106	188	219	255	152	172	196	245	202
SIG16-1 1900	102	157	131	137	145	182	169	210	186	217
SIG16-1 1910	122	94	133	162	81	116	120	134	165	147
SIG16-1 1920	153	166	108	142	128	132	130	110	153	148
SIG16-1 1930	94	39	92	152	113	122	86	123	173	173
SIG16-1 1940	149	168	97	84	58	137	94	95	121	136
SIG16-1 1950	94	85	71	78	54	82	100	63	128	80
SIG16-1 1960	90	90	99	90	54	53	35	54	57	66
SIG16-1 1970	89	67	57	70	53	66	44	14	77	37
SIG16-1 1980	41	44	99900							
SIG16-2 1850	207	312	251	203	198	297	188	301	233	200
SIG16-2 1860	26	88	248	182	149	151	207	267	222	169
SIG16-2 1870	20	34	112	159	59	78	179	112	274	164
SIG16-2 1880	216	132	179	135	135	135	257	117	165	195
SIG16-2 1890	42	48	122	186	201	112	105	138	178	107
SIG16-2 1900	31	125	91	100	89	101	160	187	234	150
SIG16-2 1910	81	89	101	110	71	92	92	138	169	120
SIG16-2 1920	132	128	120	126	120	92	107	110	117	143
SIG16-2 1930	89	48	61	70	87	97	46	86	147	129
SIG16-2 1940	118	88	71	71	71	129	106	115	135	93
SIG16-2 1950	103	93	79	115	86	110	89	72	126	77
SIG16-2 1960	85	90	61	77	49	39	27	32	32	34
SIG16-2 1970	53	41	29	37	34	68	35	12	71	39
SIG16-2 1980	41	30	99900							

Input file SIGNAL.RAD

SIGNAL KNOB PITCH PINE				
SIG01-1	15405	82	1900	1981 1947
SIG01-2	13677	92	1890	1981 1953
SIG02-1	15143	132	1850	1981 1914
SIG02-2	12673	139	1843	1981 1928
SIG03-1	14251	92	1890	1981 1941
SIG03-2	13348	102	1880	1981 1941
SIG05-1	19219	72	1910	1981 1939
SIG05-2	16211	102	1880	1981 1937
SIG06-1	13322	100	1882	1981 1939
SIG06-2	12640	102	1880	1981 1946
SIG08-1	15913	95	1887	1981 1938
SIG08-2	13339	92	1890	1981 1950
SIG09-1	11038	92	1890	1981 1961
SIG09-2	15445	92	1890	1981 1940
SIG11-1	16200	119	1863	1981 1919
SIG11-2	12150	112	1870	1981 1948
SIG12-1	14397	100	1882	1981 1939
SIG12-2	9561	82	1900	1981 0
SIG13-1	15625	112	1870	1981 1914
SIG13-2	14436	112	1870	1981 1919
SIG14-1	16196	102	1880	1981 1926
SIG14-2	17786	102	1880	1981 1920
SIG16-1	16054	120	1862	1981 1918
SIG16-2	15544	132	1850	1981 1915

Input file KNOB.RAD

SIGNAL KNOB PITCH PINE	
SIG01-1	17500
SIG01-2	19500
SIG02-1	19500
SIG02-2	16500
SIG03-1	17500
SIG03-2	13500
SIG05-1	19500
SIG05-2	17200
SIG06-1	15000
SIG06-2	14000
SIG08-1	19000
SIG08-2	16500
SIG09-1	16500
SIG09-2	19000
SIG11-1	18200
SIG11-2	17000
SIG12-1	15000
SIG12-2	12200
SIG13-1	20500
SIG13-2	18700
SIG14-1	17200
SIG14-2	21000
SIG16-1	21000
SIG16-2	18000

Output file SIGNAL.CRN

SIGNAL KNOB PITCH PINE

NUMBER OF CORES: MS= 0.220

1900	23	23	23	23	23	23	23	23	23	23
1910	24	24	24	24	24	24	24	24	24	24
1920	24	24	24	24	24	24	24	24	24	24
1930	24	24	24	24	24	24	24	24	24	24
1940	24	24	24	24	24	24	24	24	24	24
1950	24	24	24	24	24	24	24	24	24	24
1960	24	24	24	24	24	24	24	24	24	24
1970	24	24	24	24	24	24	24	24	24	24
1980	24	24								

BASAL AREA INCREMENT (not smoothed):

1900	2.061	3.388	3.261	4.532	4.618	3.648	4.722	6.217	7.493	6.535
1910	4.367	4.000	5.528	6.516	4.480	5.317	6.256	7.394	7.454	7.429
1920	8.519	8.913	7.676	7.251	6.829	6.891	7.246	7.291	10.715	11.098
1930	6.316	3.636	4.458	7.790	7.983	8.837	6.145	8.587	11.343	13.285
1940	11.410	11.571	7.657	6.777	4.730	10.630	9.499	10.736	14.176	13.039
1950	13.057	11.369	9.542	10.228	9.504	10.669	10.364	7.588	12.217	7.884
1960	9.274	8.900	8.098	10.198	6.549	5.953	5.108	8.934	7.332	5.169
1970	6.037	3.374	3.713	7.126	6.426	5.978	4.916	3.300	5.664	3.039
1980	4.030	2.409								

MEAN BAI TREND:

1900	4.431	4.554	4.677	4.801	4.926	5.051	5.177	5.303	5.429	5.555
1910	5.531	5.680	5.829	5.978	6.127	6.276	6.424	6.571	6.717	6.860
1920	7.002	7.142	7.279	7.414	7.547	7.678	7.808	7.937	8.064	8.189
1930	8.313	8.436	8.559	8.680	8.800	8.918	9.033	9.143	9.247	9.345
1940	9.435	9.516	9.587	9.650	9.703	9.745	9.776	9.793	9.796	9.784
1950	9.756	9.712	9.652	9.576	9.484	9.378	9.257	9.122	8.973	8.813
1960	8.640	8.456	8.263	8.060	7.849	7.631	7.407	7.177	6.943	6.705
1970	6.464	6.221	5.975	5.728	5.479	5.229	4.977	4.724	4.470	4.216
1980	3.961	3.706								

UPPER 95% CONFIDENCE LIMIT:

1900	5.545	5.647	5.751	5.856	5.963	6.072	6.182	6.294	6.407	6.521
1910	6.495	6.622	6.753	6.887	7.025	7.167	7.312	7.461	7.612	7.764
1920	7.919	8.074	8.230	8.386	8.543	8.700	8.857	9.014	9.172	9.329
1930	9.485	9.642	9.800	9.957	10.114	10.269	10.421	10.569	10.712	10.849
1940	10.977	11.096	11.205	11.304	11.392	11.468	11.531	11.577	11.606	11.616
1950	11.606	11.574	11.523	11.450	11.358	11.247	11.117	10.968	10.803	10.621
1960	10.423	10.210	9.984	9.745	9.495	9.235	8.966	8.689	8.406	8.117
1970	7.825	7.530	7.233	6.935	6.638	6.341	6.045	5.752	5.462	5.176
1980	4.895	4.620								

LOWER 95% CONFIDENCE LIMIT:

1900	3.317	3.460	3.604	3.747	3.889	4.031	4.171	4.312	4.450	4.588
1910	4.566	4.737	4.905	5.069	5.229	5.385	5.536	5.681	5.822	5.956
1920	6.086	6.209	6.328	6.442	6.551	6.657	6.760	6.859	6.955	7.049
1930	7.140	7.230	7.317	7.403	7.487	7.568	7.644	7.716	7.782	7.841
1940	7.892	7.935	7.970	7.996	8.013	8.022	8.021	8.009	7.986	7.952
1950	7.907	7.849	7.781	7.701	7.610	7.508	7.397	7.275	7.144	7.005
1960	6.857	6.702	6.541	6.374	6.202	6.027	5.847	5.665	5.480	5.293
1970	5.103	4.912	4.717	4.520	4.320	4.116	3.908	3.696	3.478	3.256
1980	3.027	2.793								

INDEX VALUE:

1900	0.465	0.744	0.697	0.944	0.937	0.722	0.912	1.172	1.380	1.177
1910	0.790	0.704	0.948	1.090	0.731	0.947	0.974	1.125	1.110	1.083
1920	1.217	1.248	1.055	0.978	0.905	0.897	0.928	0.919	1.329	1.355
1930	0.760	0.431	0.521	0.898	0.907	0.911	0.680	0.939	1.227	1.422
1940	1.209	1.216	0.799	0.702	0.488	1.091	0.972	1.096	1.447	1.333
1950	1.338	1.171	0.989	1.068	1.002	1.138	1.120	0.832	1.361	0.895
1960	1.073	1.052	0.980	1.265	0.834	0.780	0.690	1.245	1.056	0.771
1970	0.934	0.542	0.621	1.244	1.173	1.143	0.988	0.699	1.267	0.721
1980	1.017	0.650								

SIGNAL KNOB PITCH PINE

NUMBER OF CORES: MS= 0.221

1900	23	23	23	23	23	23	23	23	23	23
1910	24	24	24	24	24	24	24	24	24	24
1920	24	24	24	24	24	24	24	24	24	24
1930	24	24	24	24	24	24	24	24	24	24
1940	24	24	24	24	24	24	24	24	24	24
1950	24	24	24	24	24	24	24	24	24	24
1960	24	24	24	24	24	24	24	24	24	24
1970	24	24	24	24	24	24	24	24	24	24
1980	24	24	24	24	24	24	24	24	24	24

BASAL AREA INCREMENT (not smoothed):

1900	3.725	5.671	5.417	7.383	7.414	5.653	7.162	9.490	11.442	9.820
1910	6.609	5.962	8.149	9.673	6.488	7.499	8.785	10.282	10.292	10.220
1920	11.657	12.045	10.290	9.765	9.209	9.208	9.656	9.624	13.986	14.480
1930	8.222	4.612	5.703	10.016	10.206	11.372	7.980	11.114	14.462	16.872
1940	14.415	14.574	9.605	8.591	5.897	13.476	11.880	13.365	17.554	15.801
1950	15.656	13.796	11.577	12.485	11.493	12.923	12.504	9.129	14.722	9.572
1960	11.246	10.731	9.662	12.070	7.770	7.085	6.078	10.703	8.758	6.159
1970	7.238	3.972	4.439	8.523	7.623	7.123	5.778	3.896	6.667	3.623
1980	4.768	2.861								

MEAN BAI TREND:

1900	7.531	7.611	7.695	7.781	7.871	7.964	8.059	8.158	8.258	8.360
1910	8.227	8.356	8.487	8.620	8.754	8.889	9.025	9.160	9.295	9.429
1920	9.561	9.691	9.819	9.945	10.070	10.193	10.315	10.436	10.556	10.675
1930	10.792	10.910	11.027	11.144	11.259	11.373	11.482	11.587	11.685	11.775
1940	11.856	11.926	11.985	12.034	12.071	12.095	12.106	12.101	12.079	12.040
1950	11.982	11.905	11.809	11.696	11.564	11.415	11.250	11.069	10.873	10.662
1960	10.439	10.202	9.955	9.697	9.431	9.156	8.875	8.589	8.297	8.002
1970	7.703	7.401	7.098	6.793	6.486	6.177	5.867	5.556	5.244	4.931
1980	4.618	4.305								

UPPER 95% CONFIDENCE LIMIT:

1900	8.927	8.970	9.016	9.068	9.126	9.188	9.256	9.329	9.406	9.488
1910	9.394	9.491	9.593	9.701	9.815	9.935	10.060	10.191	10.325	10.463
1920	10.604	10.747	10.892	11.038	11.185	11.334	11.484	11.635	11.786	11.938
1930	12.089	12.242	12.396	12.550	12.704	12.856	13.004	13.149	13.286	13.416
1940	13.535	13.644	13.740	13.825	13.896	13.954	13.995	14.018	14.021	14.002
1950	13.960	13.894	13.806	13.695	13.562	13.407	13.232	13.036	12.822	12.589
1960	12.339	12.074	11.793	11.499	11.192	10.875	10.549	10.215	9.875	9.528
1970	9.178	8.825	8.471	8.116	7.762	7.408	7.055	6.705	6.359	6.016
1980	5.679	5.347								

LOWER 95% CONFIDENCE LIMIT:

1900	6.134	6.253	6.373	6.494	6.616	6.739	6.863	6.987	7.110	7.233
1910	7.059	7.222	7.382	7.539	7.693	7.843	7.989	8.130	8.265	8.394
1920	8.517	8.634	8.746	8.852	8.954	9.052	9.146	9.238	9.326	9.412
1930	9.495	9.577	9.658	9.738	9.815	9.890	9.960	10.025	10.084	10.135
1940	10.176	10.209	10.231	10.243	10.245	10.236	10.216	10.184	10.138	10.078
1950	10.004	9.915	9.813	9.696	9.567	9.424	9.269	9.102	8.924	8.736
1960	8.538	8.331	8.117	7.896	7.669	7.437	7.201	6.962	6.720	6.475
1970	6.227	5.977	5.725	5.469	5.210	4.947	4.679	4.406	4.129	3.846
1980	3.557	3.262								

INDEX VALUE:

1900	0.495	0.745	0.704	0.949	0.942	0.710	0.889	1.163	1.386	1.175
1910	0.803	0.714	0.960	1.122	0.741	0.844	0.973	1.122	1.107	1.084
1920	1.219	1.243	1.048	0.982	0.915	0.903	0.936	0.922	1.325	1.356
1930	0.762	0.423	0.517	0.899	0.906	1.000	0.695	0.959	1.238	1.433
1940	1.216	1.222	0.801	0.714	0.489	1.114	0.981	1.104	1.453	1.312
1950	1.307	1.159	0.980	1.068	0.994	1.132	1.111	0.825	1.354	0.898
1960	1.077	1.052	0.971	1.245	0.824	0.774	0.685	1.246	1.055	0.770
1970	0.940	0.537	0.625	1.255	1.175	1.153	0.985	0.701	1.271	0.735
1980	1.032	0.665								

Output file KNOB.JND

SIGNAL KNOB PITCH PINE

RING WIDTHS	SIG01-1	MS=	0.344	1900=	0.00	CM.				
1900	0.119	0.177	0.133	0.349	0.235	0.038	0.093	0.210	0.274	0.284
1910	0.140	0.109	0.204	0.310	0.183	0.094	0.162	0.160	0.191	0.293
1920	0.244	0.231	0.178	0.198	0.163	0.209	0.207	0.244	0.312	0.312
1930	0.131	0.107	0.111	0.215	0.168	0.204	0.179	0.248	0.322	0.346
1940	0.240	0.313	0.185	0.187	0.107	0.317	0.285	0.288	0.324	0.263
1950	0.247	0.258	0.213	0.180	0.197	0.255	0.229	0.176	0.214	0.178
1960	0.232	0.212	0.144	0.181	0.141	0.105	0.119	0.255	0.172	0.116
1970	0.116	0.059	0.044	0.139	0.106	0.077	0.080	0.058	0.102	0.074
1980	0.088	0.042								

AREA	SIG01-1									
1900	1.611	2.561	2.054	5.917	4.416	0.747	1.865	4.412	6.174	6.897
1910	3.586	2.877	5.586	8.989	5.590	2.953	5.220	5.317	6.558	10.506
1920	9.160	9.017	7.177	8.217	6.949	9.155	9.338	11.353	15.061	15.673
1930	6.763	5.604	5.889	11.628	9.288	11.517	10.321	14.632	19.574	21.759
1940	15.535	20.804	12.586	12.940	7.503	22.651	20.904	21.642	24.971	20.754
1950	19.887	21.182	17.803	15.267	16.942	22.292	20.368	15.878	19.568	16.495
1960	21.798	20.215	13.892	17.646	13.889	10.424	11.898	25.795	17.630	11.995
1970	12.079	6.176	4.620	14.676	11.273	8.233	8.593	6.255	11.052	8.059
1980	9.629	4.613								

SPLINE	SIG01-1									
1900	1.767	2.069	2.371	2.673	2.975	3.277	3.580	3.885	4.191	4.498
1910	4.807	5.119	5.433	5.750	6.071	6.396	6.726	7.060	7.399	7.743
1920	8.091	8.443	8.799	9.160	9.525	9.895	10.269	10.646	11.027	11.409
1930	11.794	12.182	12.573	12.965	13.357	13.747	14.132	14.511	14.880	15.235
1940	15.576	15.898	16.202	16.485	16.746	16.983	17.193	17.374	17.523	17.640
1950	17.723	17.772	17.787	17.769	17.718	17.634	17.517	17.369	17.189	16.979
1960	16.740	16.473	16.179	15.860	15.519	15.156	14.775	14.376	13.960	13.530
1970	13.088	12.636	12.177	11.713	11.244	10.772	10.297	9.821	9.343	8.865
1980	8.385	7.906								

INDEX	SIG01-1									
1900	0.912	1.238	0.866	2.214	1.484	0.228	0.521	1.136	1.473	1.533
1910	0.746	0.562	1.128	1.563	0.921	0.462	0.776	0.753	0.886	1.357
1920	1.132	1.068	0.816	0.897	0.730	0.925	0.909	1.066	1.366	1.374
1930	0.573	0.460	0.468	0.897	0.695	0.838	0.730	1.008	1.315	1.428
1940	0.997	1.309	0.777	0.785	0.448	1.334	1.216	1.246	1.425	1.177
1950	1.122	1.192	1.001	0.859	0.956	1.264	1.163	0.914	1.138	0.971
1960	1.302	1.227	0.859	1.113	0.895	0.688	0.805	1.794	1.263	0.887
1970	0.923	0.489	0.379	1.253	1.003	0.764	0.835	0.637	1.183	0.909
1980	1.148	0.583								

RING WIDTHS	SIG01-2	MS=	0.399	1890=	0.00	CM.				
1890	0.115	0.133	0.176	0.133	0.145	0.106	0.134	0.162	0.263	0.155
1900	0.098	0.128	0.089	0.156	0.177	0.013	0.043	0.100	0.166	0.167
1910	0.078	0.048	0.095	0.191	0.127	0.054	0.070	0.099	0.093	0.158
1920	0.149	0.142	0.140	0.174	0.161	0.193	0.176	0.164	0.227	0.203
1930	0.106	0.051	0.066	0.141	0.157	0.241	0.199	0.259	0.221	0.311
1940	0.259	0.240	0.122	0.172	0.102	0.287	0.247	0.243	0.345	0.221
1950	0.177	0.173	0.171	0.186	0.206	0.202	0.156	0.117	0.202	0.183
1960	0.253	0.210	0.107	0.145	0.110	0.103	0.086	0.220	0.170	0.122
1970	0.101	0.048	0.062	0.149	0.123	0.118	0.069	0.044	0.096	0.054
1980	0.080	0.043								

AREA	SIG01-2									
1890	4.249	5.018	6.811	5.276	5.879	4.381	5.639	6.968	11.664	7.078
1900	4.553	6.037	4.259	7.585	8.791	0.653	2.169	5.089	8.586	8.812
1910	4.176	2.589	5.166	10.559	7.148	3.070	4.007	5.719	5.429	9.348
1920	8.959	8.668	8.670	10.947	10.299	12.560	11.658	11.038	15.557	14.187
1930	7.511	3.639	4.733	10.204	11.509	17.967	15.111	20.040	17.433	25.052
1940	21.327	20.139	10.376	14.787	8.857	25.272	22.164	22.179	32.126	20.972
1950	17.018	16.824	16.814	18.498	20.740	20.597	16.082	12.162	21.199	19.427
1960	27.204	22.886	11.768	16.062	12.273	11.561	9.704	25.035	19.553	14.144
1970	11.780	5.621	7.282	17.599	14.633	14.128	8.302	5.309	11.626	6.565
1980	9.760	5.263								

Output file KNOB.IND, continued

SPLINE	SIG01-2									
1890	4.902	4.959	5.017	5.074	5.132	5.189	5.247	5.306	5.366	5.426
1900	5.490	5.557	5.628	5.707	5.792	5.887	5.992	6.108	6.237	6.378
1910	6.534	6.704	6.889	7.090	7.307	7.541	7.791	8.059	8.342	8.642
1920	8.956	9.283	9.623	9.974	10.336	10.706	11.085	11.471	11.863	12.260
1930	12.662	13.069	13.480	13.893	14.305	14.712	15.111	15.498	15.871	16.226
1940	16.560	16.873	17.162	17.428	17.669	17.882	18.066	18.218	18.338	18.423
1950	18.476	18.496	18.484	18.441	18.368	18.265	18.133	17.973	17.784	17.569
1960	17.326	17.057	16.764	16.449	16.114	15.761	15.391	15.006	14.605	14.191
1970	13.765	13.330	12.886	12.436	11.979	11.516	11.049	10.578	10.106	9.632
1980	9.157	8.681								

INDEX	SIG01-2									
1890	0.867	1.012	1.358	1.040	1.146	0.844	1.075	1.313	2.174	1.304
1900	0.829	1.087	0.757	1.329	1.518	0.111	0.362	0.833	1.311	1.382
1910	0.639	0.386	0.750	1.489	0.978	0.407	0.514	0.710	0.651	1.082
1920	1.000	0.934	0.901	1.098	0.996	1.173	1.052	0.962	1.311	1.157
1930	0.593	0.278	0.351	0.734	0.805	1.221	1.000	1.293	1.098	1.544
1940	1.288	1.194	0.605	0.848	0.501	1.413	1.227	1.217	1.752	1.138
1950	0.921	0.910	0.910	1.003	1.129	1.128	0.887	0.677	1.192	1.106
1960	1.570	1.342	0.702	0.976	0.762	0.734	0.630	1.668	1.339	0.997
1970	0.856	0.422	0.565	1.415	1.222	1.227	0.751	0.502	1.150	0.682
1980	1.066	0.606								

RING WIDTHS	SIG02-1 MS= 0.533 1850= 0.00 CM.									
1850	0.284	0.269	0.267	0.302	0.251	0.212	0.148	0.199	0.233	0.206
1860	0.089	0.161	0.250	0.148	0.190	0.314	0.136	0.304	0.254	0.191
1870	0.032	0.128	0.112	0.108	0.025	0.049	0.067	0.102	0.231	0.142
1880	0.209	0.095	0.190	0.149	0.193	0.107	0.183	0.151	0.259	0.324
1890	0.090	0.112	0.143	0.166	0.208	0.049	0.075	0.169	0.093	0.106
1900	0.070	0.148	0.134	0.116	0.117	0.025	0.111	0.150	0.192	0.141
1910	0.049	0.055	0.107	0.103	0.091	0.072	0.121	0.150	0.162	0.136
1920	0.147	0.162	0.129	0.198	0.185	0.112	0.096	0.103	0.165	0.219
1930	0.082	0.000	0.083	0.111	0.089	0.097	0.086	0.073	0.137	0.157
1940	0.119	0.087	0.077	0.073	0.048	0.099	0.073	0.081	0.117	0.095
1950	0.107	0.085	0.049	0.064	0.034	0.049	0.051	0.032	0.062	0.045
1960	0.050	0.027	0.034	0.057	0.023	0.027	0.025	0.072	0.030	0.004
1970	0.026	0.004	0.005	0.049	0.023	0.026	0.004	0.014	0.015	0.025
1980	0.000	0.000								

AREA	SIG02-1									
1850	8.028	8.071	8.461	10.110	8.839	7.774	5.594	7.739	9.378	8.575
1860	3.787	6.977	11.157	6.790	8.919	15.237	6.792	15.602	13.481	10.404
1870	1.766	7.126	6.320	6.169	1.438	2.831	3.895	5.984	13.794	8.646
1880	12.955	5.979	12.129	9.670	12.733	7.160	12.413	10.401	18.173	23.328
1890	6.597	8.281	10.687	12.567	15.991	3.807	5.856	13.325	7.409	8.511
1900	5.659	12.067	11.044	9.651	9.820	2.109	9.414	12.844	16.647	12.372
1910	4.329	4.877	9.542	9.254	8.231	6.549	11.080	13.863	15.131	12.830
1920	13.998	15.584	12.527	19.431	18.378	11.231	9.689	10.460	16.895	22.638
1930	8.573	0.000	8.720	11.730	9.461	10.368	9.242	7.881	14.881	17.198
1940	13.139	9.662	8.591	8.179	5.396	11.176	8.280	9.227	13.400	10.944
1950	12.394	9.897	5.726	7.502	3.996	5.771	6.023	3.787	7.356	5.354
1960	5.964	3.227	4.070	6.840	2.766	3.251	3.014	8.704	3.636	0.485
1970	3.156	0.486	0.608	5.963	2.804	3.174	0.489	1.711	1.835	3.061
1980	0.000	0.000								

SPLINE	SIG02-1									
1850	8.023	8.044	8.065	8.087	8.108	8.130	8.153	8.176	8.201	8.227
1860	8.254	8.282	8.311	8.341	8.371	8.401	8.430	8.460	8.490	8.522
1870	8.558	8.598	8.645	8.700	8.763	8.834	8.914	9.001	9.092	9.188
1880	9.284	9.381	9.476	9.568	9.657	9.740	9.816	9.885	9.945	9.996
1890	10.038	10.073	10.102	10.125	10.145	10.162	10.179	10.197	10.218	10.242
1900	10.270	10.303	10.341	10.385	10.435	10.492	10.555	10.625	10.699	10.779
1910	10.863	10.952	11.046	11.144	11.243	11.343	11.442	11.537	11.625	11.705
1920	11.774	11.831	11.874	11.903	11.915	11.913	11.895	11.862	11.816	11.756
1930	11.683	11.599	11.507	11.405	11.295	11.177	11.050	10.914	10.769	10.614
1940	10.449	10.273	10.089	9.897	9.696	9.488	9.272	9.049	8.818	8.579
1950	8.334	8.081	7.823	7.562	7.298	7.032	6.766	6.500	6.234	5.970
1960	5.706	5.445	5.185	4.927	4.671	4.417	4.166	3.916	3.667	3.421

Output file KNOB.IND, continued

1970	3.176	2.933	2.692	2.452	2.213	1.975	1.736	1.498	1.260	1.021
1980	0.783	0.544								

INDEX	SIG02-1									
1850	1.001	1.003	1.049	1.250	1.090	0.956	0.686	0.947	1.143	1.042
1860	0.459	0.843	1.342	0.814	1.065	1.814	0.806	1.844	1.588	1.221
1870	0.206	0.829	0.731	0.709	0.164	0.320	0.437	0.665	1.517	0.941
1880	1.395	0.637	1.280	1.011	1.319	0.735	1.265	1.052	1.827	2.334
1890	0.657	0.822	1.058	1.241	1.576	0.375	0.575	1.307	0.725	0.831
1900	0.551	1.171	1.068	0.929	0.941	0.201	0.892	1.209	1.556	1.148
1910	0.399	0.445	0.864	0.830	0.732	0.577	0.968	1.202	1.302	1.096
1920	1.189	1.317	1.055	1.632	1.542	0.943	0.815	0.882	1.430	1.930
1930	0.734	0.000	0.758	1.028	0.838	0.928	0.836	0.722	1.382	1.620
1940	1.257	0.940	0.852	0.826	0.557	1.178	0.893	1.020	1.520	1.276
1950	1.487	1.225	0.732	0.992	0.548	0.821	0.890	0.583	1.180	0.897
1960	1.045	0.593	0.785	1.388	0.592	0.736	0.724	2.223	0.991	0.142
1970	0.994	0.166	0.226	2.432	1.267	1.607	0.281	1.142	1.457	2.998
1980	0.000	0.000								

RING WIDTHS	SIG02-2 MS= 0.437 1843= 0.00 CM.									
1843	0.142	0.218	0.187	0.219	0.290	0.306	0.188			
1850	0.192	0.207	0.280	0.300	0.229	0.178	0.156	0.208	0.225	0.141
1860	0.052	0.125	0.152	0.108	0.156	0.269	0.100	0.203	0.141	0.156
1870	0.033	0.100	0.060	0.080	0.021	0.025	0.044	0.103	0.186	0.174
1880	0.171	0.094	0.122	0.133	0.128	0.109	0.136	0.112	0.148	0.236
1890	0.072	0.119	0.088	0.112	0.124	0.039	0.052	0.072	0.045	0.041
1900	0.035	0.075	0.094	0.046	0.062	0.013	0.056	0.084	0.095	0.080
1910	0.036	0.040	0.063	0.042	0.056	0.048	0.083	0.116	0.117	0.098
1920	0.084	0.113	0.078	0.057	0.060	0.049	0.046	0.052	0.068	0.103
1930	0.063	0.000	0.045	0.048	0.051	0.087	0.051	0.056	0.096	0.103
1940	0.091	0.080	0.063	0.059	0.039	0.071	0.059	0.072	0.102	0.062
1950	0.097	0.073	0.049	0.055	0.045	0.051	0.061	0.057	0.076	0.055
1960	0.075	0.053	0.049	0.057	0.020	0.014	0.014	0.027	0.039	0.017
1970	0.024	0.014	0.018	0.055	0.025	0.034	0.006	0.018	0.017	0.024
1980	0.000	0.000								

AREA	SIG02-2									
1843	3.478	5.586	5.029	6.169	8.633	9.682	6.240			
1850	6.602	7.378	10.408	11.698	9.310	7.464	6.705	9.178	10.235	6.576
1860	2.457	5.975	7.398	5.345	7.849	13.894	5.281	10.914	7.733	8.701
1870	1.860	5.679	3.437	4.618	1.219	1.455	2.570	6.064	11.119	10.598
1880	10.601	5.906	7.748	8.553	8.336	7.180	9.063	7.551	10.099	16.388
1890	5.069	8.450	6.306	8.096	9.056	2.868	3.839	5.344	3.356	3.069
1900	2.628	5.658	7.141	3.515	4.758	1.001	4.323	6.522	7.429	6.300
1910	2.848	3.174	5.020	3.360	4.498	3.871	6.728	9.475	9.642	8.143
1920	7.027	9.524	6.621	4.862	5.140	4.215	3.970	4.504	5.916	9.016
1930	5.548	0.000	3.978	4.257	4.539	7.781	4.583	5.051	8.705	9.404
1940	8.364	7.396	5.853	5.504	3.650	6.670	5.566	6.822	9.721	5.941
1950	9.343	7.070	4.765	5.366	4.404	5.007	6.010	5.637	7.548	5.485
1960	7.510	5.329	4.942	5.768	2.029	1.422	1.423	2.747	3.977	1.736
1970	2.455	1.433	1.845	5.650	2.574	3.507	0.620	1.860	1.759	2.486
1980	0.000	0.000								

SPLINE	SIG02-2									
1843	6.971	7.017	7.063	7.107	7.150	7.190	7.226			
1850	7.258	7.286	7.308	7.325	7.335	7.339	7.338	7.332	7.321	7.306
1860	7.288	7.268	7.247	7.223	7.199	7.172	7.145	7.117	7.089	7.062
1870	7.038	7.017	7.000	6.988	6.981	6.979	6.981	6.986	6.991	6.994
1880	6.994	6.990	6.981	6.965	6.943	6.913	6.874	6.828	6.773	6.709
1890	6.637	6.558	6.473	6.384	6.293	6.201	6.110	6.021	5.937	5.858
1900	5.786	5.720	5.662	5.612	5.569	5.535	5.509	5.491	5.480	5.475
1910	5.478	5.486	5.500	5.520	5.543	5.570	5.599	5.629	5.658	5.686
1920	5.713	5.738	5.761	5.783	5.805	5.828	5.851	5.876	5.901	5.928
1930	5.956	5.986	6.017	6.049	6.081	6.113	6.144	6.172	6.197	6.216
1940	6.230	6.238	6.239	6.233	6.220	6.200	6.172	6.136	6.090	6.035
1950	5.971	5.897	5.813	5.720	5.618	5.508	5.389	5.263	5.128	4.985
1960	4.834	4.677	4.513	4.345	4.172	3.996	3.818	3.638	3.457	3.275
1970	3.092	2.908	2.724	2.538	2.352	2.164	1.975	1.786	1.595	1.405
1980	1.213	1.022								

Output file KNOB.IND, continued

INDEX	SIG02-2									
1843	0.499	0.796	0.712	0.868	1.207	1.347	0.864			
1850	0.910	1.013	1.424	1.597	1.269	1.017	0.914	1.252	1.398	0.900
1860	0.337	0.822	1.021	0.740	1.090	1.937	0.739	1.533	1.091	1.232
1870	0.264	0.809	0.491	0.661	0.175	0.208	0.368	0.868	1.591	1.515
1880	1.516	0.845	1.110	1.228	1.201	1.039	1.318	1.106	1.491	2.443
1890	0.764	1.289	0.974	1.268	1.439	0.463	0.628	0.887	0.565	0.524
1900	0.454	0.985	1.261	0.626	0.854	0.181	0.785	1.188	1.356	1.151
1910	0.520	0.579	0.913	0.609	0.811	0.695	1.201	1.683	1.704	1.432
1920	1.230	1.660	1.149	0.841	0.885	0.723	0.679	0.767	1.002	1.521
1930	0.931	0.000	0.661	0.704	0.746	1.273	0.746	0.818	1.405	1.513
1940	1.343	1.186	0.938	0.883	0.587	1.076	0.902	1.112	1.596	0.984
1950	1.565	1.199	0.820	0.938	0.784	0.909	1.115	1.071	1.472	1.100
1960	1.554	1.139	1.095	1.328	0.486	0.356	0.373	0.755	1.150	0.530
1970	0.794	0.493	0.677	2.226	1.095	1.621	0.314	1.042	1.103	1.770
1980	0.000	0.000								

RING WIDTHS	SIG03-1 MS= 0.486 1890= 0.00 CM.									
1890	0.388	0.175	0.147	0.216	0.211	0.310	0.071	0.108	0.248	0.320
1900	0.180	0.048	0.180	0.171	0.232	0.228	0.092	0.171	0.208	0.203
1910	0.040	0.052	0.156	0.228	0.124	0.211	0.235	0.243	0.221	0.201
1920	0.235	0.197	0.184	0.217	0.192	0.187	0.190	0.168	0.312	0.286
1930	0.110	0.035	0.060	0.191	0.232	0.217	0.132	0.189	0.281	0.357
1940	0.263	0.273	0.111	0.124	0.045	0.155	0.180	0.180	0.250	0.200
1950	0.092	0.175	0.090	0.188	0.143	0.115	0.172	0.118	0.137	0.067
1960	0.084	0.084	0.125	0.127	0.070	0.089	0.088	0.101	0.054	0.043
1970	0.069	0.030	0.029	0.112	0.054	0.117	0.084	0.050	0.061	0.012
1980	0.071	0.029								

AREA	SIG03-1									
1890	8.394	4.095	3.589	5.520	5.675	8.845	2.111	3.271	7.790	10.622
1900	6.258	1.703	6.516	6.378	8.947	9.123	3.774	7.155	8.951	8.998
1910	1.804	2.360	7.181	10.770	5.995	10.422	11.937	12.708	11.880	11.071
1920	13.266	11.388	10.857	13.078	11.818	11.733	12.146	10.928	20.766	19.573
1930	7.665	2.455	4.226	13.604	16.832	16.050	9.908	14.377	21.790	28.399
1940	21.433	22.708	9.367	10.555	3.854	13.374	15.721	15.924	22.455	18.246
1950	8.478	16.273	8.444	17.802	13.690	11.103	16.761	11.606	13.585	6.687
1960	8.423	8.467	12.682	12.986	7.201	9.200	9.145	10.556	5.670	4.528
1970	7.291	3.179	3.079	11.939	5.785	12.596	9.096	5.436	6.653	1.311
1980	7.778	3.186								

SPLINE	SIG03-1									
1890	4.536	4.704	4.873	5.042	5.213	5.385	5.559	5.735	5.914	6.096
1900	6.282	6.471	6.665	6.864	7.069	7.279	7.494	7.717	7.946	8.181
1910	8.424	8.674	8.931	9.193	9.459	9.729	9.999	10.270	10.538	10.804
1920	11.066	11.324	11.577	11.824	12.064	12.298	12.524	12.742	12.951	13.151
1930	13.341	13.523	13.695	13.858	14.008	14.143	14.263	14.363	14.443	14.500
1940	14.533	14.542	14.528	14.493	14.438	14.364	14.272	14.159	14.027	13.876
1950	13.706	13.518	13.314	13.095	12.861	12.614	12.355	12.084	11.804	11.516
1960	11.222	10.923	10.620	10.314	10.005	9.694	9.382	9.070	8.760	8.450
1970	8.143	7.838	7.534	7.232	6.931	6.629	6.326	6.023	5.718	5.414
1980	5.109	4.804								

INDEX	SIG03-1									
1890	1.851	0.871	0.737	1.095	1.089	1.643	0.380	0.570	1.317	1.742
1900	0.996	0.263	0.978	0.929	1.266	1.253	0.504	0.927	1.127	1.100
1910	0.214	0.272	0.804	1.172	0.634	1.071	1.194	1.237	1.127	1.025
1920	1.199	1.006	0.938	1.106	0.980	0.954	0.970	0.858	1.603	1.488
1930	0.575	0.182	0.309	0.982	1.202	1.135	0.695	1.001	1.509	1.958
1940	1.475	1.562	0.645	0.728	0.267	0.931	1.102	1.125	1.601	1.315
1950	0.619	1.204	0.634	1.359	1.064	0.880	1.357	0.960	1.151	0.581
1960	0.751	0.775	1.194	1.259	0.720	0.949	0.975	1.164	0.647	0.536
1970	0.895	0.406	0.409	1.651	0.835	1.900	1.438	0.903	1.163	0.242
1980	1.523	0.663								

RING WIDTHS	SIG03-2 MS= 0.480 1880= 0.00 CM.									
1880	0.521	0.445	0.377	0.266	0.117	0.221	0.332	0.179	0.172	0.256
1890	0.119	0.115	0.179	0.183	0.273	0.088	0.067	0.213	0.154	0.142
1900	0.044	0.110	0.087	0.173	0.195	0.020	0.101	0.152	0.184	0.181
1910	0.043	0.035	0.081	0.111	0.098	0.141	0.206	0.191	0.191	0.208

Output file KNOB.IND, continued

1920	0.188	0.187	0.143	0.140	0.140	0.123	0.123	0.102	0.238	0.218
1930	0.100	0.021	0.055	0.121	0.137	0.124	0.061	0.098	0.177	0.293
1940	0.222	0.272	0.121	0.135	0.052	0.131	0.146	0.098	0.162	0.146
1950	0.131	0.114	0.071	0.120	0.148	0.088	0.135	0.097	0.124	0.034
1960	0.055	0.057	0.066	0.091	0.028	0.032	0.032	0.053	0.056	0.044
1970	0.062	0.040	0.033	0.074	0.047	0.056	0.062	0.031	0.033	0.009
1980	0.065	0.015								

AREA		SIG03-2								
1880	1.350	2.504	3.095	2.721	1.338	2.761	4.725	2.835	2.914	4.681
1890	2.316	2.323	3.781	4.073	6.468	2.185	1.696	5.579	4.211	4.015
1900	1.270	3.228	2.607	5.325	6.227	0.652	3.332	5.135	6.411	6.514
1910	1.578	1.293	3.021	4.207	3.779	5.543	8.323	7.955	8.184	9.173
1920	8.525	8.700	6.801	6.783	6.906	6.169	6.264	5.267	12.543	11.802
1930	5.513	1.166	3.066	6.813	7.825	7.184	3.570	5.784	10.599	17.978
1940	13.981	17.552	7.957	8.987	3.492	8.873	10.016	6.798	11.370	10.388
1950	9.435	8.298	5.209	8.877	11.072	6.649	10.295	7.468	9.632	2.658
1960	4.315	4.492	5.227	7.251	2.242	2.568	2.574	4.278	4.539	3.580
1970	5.066	3.281	2.714	6.112	3.900	4.665	5.187	2.603	2.777	0.759
1980	5.494	1.272								

SPLINE		SIG03-2								
1880	2.277	2.357	2.437	2.517	2.597	2.676	2.755	2.834	2.912	2.990
1890	3.068	3.145	3.224	3.303	3.382	3.462	3.545	3.629	3.715	3.805
1900	3.897	3.994	4.095	4.201	4.311	4.426	4.546	4.671	4.800	4.934
1910	5.073	5.216	5.364	5.516	5.671	5.827	5.984	6.140	6.294	6.445
1920	6.593	6.737	6.877	7.013	7.145	7.274	7.399	7.521	7.639	7.752
1930	7.862	7.968	8.071	8.170	8.265	8.353	8.434	8.507	8.568	8.616
1940	8.650	8.667	8.668	8.652	8.622	8.577	8.518	8.444	8.356	8.254
1950	8.139	8.010	7.870	7.718	7.556	7.384	7.204	7.015	6.820	6.621
1960	6.418	6.214	6.009	5.804	5.600	5.398	5.199	5.002	4.808	4.616
1970	4.426	4.238	4.051	3.866	3.681	3.497	3.312	3.128	2.944	2.759
1980	2.575	2.390								

INDEX		SIG03-2								
1880	0.593	1.062	1.270	1.081	0.515	1.032	1.715	1.000	1.001	1.566
1890	0.755	0.738	1.173	1.233	1.912	0.631	0.478	1.537	1.133	1.055
1900	0.326	0.808	0.636	1.267	1.444	0.147	0.733	1.099	1.335	1.320
1910	0.311	0.248	0.563	0.763	0.666	0.951	1.391	1.296	1.300	1.423
1920	1.293	1.291	0.989	0.967	0.967	0.848	0.847	0.700	1.642	1.522
1930	0.701	0.146	0.380	0.834	0.947	0.860	0.423	0.680	1.237	2.087
1940	1.616	2.025	0.918	1.039	0.405	1.034	1.176	0.805	1.361	1.259
1950	1.159	1.036	0.662	1.150	1.465	0.900	1.429	1.064	1.412	0.401
1960	0.672	0.723	0.870	1.249	0.400	0.476	0.495	0.855	0.944	0.776
1970	1.145	0.774	0.670	1.581	1.059	1.334	1.566	0.832	0.944	0.275
1980	2.134	0.532								

RING WIDTHS		SIG05-1		MS=	0.288	1910=	0.00	CM.		
1910	0.325	0.362	0.329	0.512	0.319	0.284	0.346	0.411	0.379	0.353
1920	0.529	0.527	0.499	0.380	0.309	0.319	0.278	0.379	0.438	0.385
1930	0.160	0.145	0.126	0.243	0.242	0.323	0.249	0.296	0.333	0.370
1940	0.330	0.303	0.170	0.106	0.106	0.237	0.274	0.328	0.395	0.681
1950	0.738	0.376	0.292	0.262	0.216	0.287	0.248	0.214	0.316	0.182
1960	0.191	0.209	0.249	0.360	0.209	0.165	0.126	0.256	0.213	0.145
1970	0.117	0.095	0.067	0.110	0.104	0.068	0.085	0.047	0.099	0.040
1980	0.034	0.019								

AREA		SIG05-1								
1910	0.906	1.790	2.341	4.996	3.946	4.051	5.620	7.653	7.998	8.261
1920	13.845	15.541	16.324	13.480	11.631	12.636	11.534	16.506	20.200	18.751
1930	8.067	7.449	6.580	12.973	13.288	18.309	14.562	17.817	20.702	23.820
1940	21.970	20.775	11.909	7.517	7.588	17.221	20.349	24.980	30.980	55.713
1950	63.666	33.753	26.825	24.525	20.543	27.750	24.396	21.362	32.070	18.755
1960	19.906	22.045	26.623	39.179	23.119	18.446	14.201	29.160	24.576	16.893
1970	13.727	11.210	7.940	13.097	12.452	8.179	10.264	5.695	12.041	4.883
1980	4.158	2.327								

SPLINE		SIG05-1								
1910	2.767	3.439	4.111	4.782	5.452	6.119	6.784	7.446	8.102	8.752
1920	9.395	10.030	10.656	11.273	11.881	12.481	13.075	13.663	14.247	14.826

Output file KNOB.IND, continued

1930	15.403	15.980	16.558	17.137	17.718	18.298	18.875	19.448	20.015	20.572
1940	21.118	21.649	22.165	22.662	23.138	23.587	24.002	24.375	24.699	24.964
1950	25.165	25.298	25.362	25.361	25.294	25.166	24.977	24.729	24.423	24.061
1960	23.645	23.176	22.656	22.086	21.467	20.802	20.096	19.350	18.569	17.757
1970	16.917	16.054	15.172	14.275	13.365	12.445	11.517	10.583	9.645	8.704
1980	7.762	6.819								

INDEX	SIG05-1									
1910	0.327	0.520	0.569	1.045	0.724	0.662	0.828	1.028	0.987	0.944
1920	1.474	1.549	1.532	1.196	0.979	1.012	0.882	1.208	1.418	1.265
1930	0.524	0.466	0.397	0.757	0.750	1.001	0.771	0.916	1.034	1.158
1940	1.040	0.960	0.537	0.332	0.328	0.730	0.848	1.025	1.254	2.232
1950	2.530	1.334	1.058	0.967	0.812	1.103	0.977	0.864	1.313	0.779
1960	0.842	0.951	1.175	1.774	1.077	0.887	0.707	1.507	1.323	0.951
1970	0.811	0.698	0.523	0.917	0.932	0.657	0.891	0.538	1.248	0.561
1980	0.536	0.341								

RING WIDTHS	SIG05-2 MS= 0.461 1880= 0.00 CM.									
1880	0.284	0.350	0.205	0.139	0.223	0.243	0.291	0.210	0.264	0.349
1890	0.048	0.019	0.110	0.167	0.178	0.099	0.116	0.191	0.208	0.212
1900	0.029	0.069	0.069	0.127	0.157	0.015	0.051	0.137	0.128	0.176
1910	0.114	0.096	0.114	0.116	0.110	0.120	0.171	0.256	0.161	0.211
1920	0.206	0.199	0.157	0.203	0.161	0.236	0.209	0.219	0.302	0.259
1930	0.180	0.206	0.182	0.285	0.222	0.208	0.138	0.193	0.241	0.218
1940	0.240	0.266	0.259	0.153	0.136	0.167	0.182	0.180	0.286	0.166
1950	0.184	0.206	0.173	0.168	0.173	0.227	0.209	0.118	0.183	0.110
1960	0.177	0.158	0.145	0.137	0.113	0.126	0.096	0.144	0.108	0.068
1970	0.065	0.028	0.043	0.084	0.100	0.071	0.047	0.031	0.088	0.019
1980	0.017	0.006								

AREA	SIG05-2									
1880	2.018	3.184	2.223	1.657	2.912	3.529	4.715	3.733	5.086	7.395
1890	1.077	0.430	2.536	3.995	4.451	2.562	3.080	5.256	5.984	6.379
1900	0.895	2.150	2.180	4.090	5.196	0.505	1.726	4.718	4.514	6.375
1910	4.233	3.628	4.384	4.544	4.387	4.873	7.100	10.973	7.112	9.567
1920	9.610	9.537	7.700	10.185	8.262	12.405	11.278	12.112	17.197	15.205
1930	10.815	12.628	11.378	18.236	14.558	13.921	9.386	13.328	16.971	15.666
1940	17.592	19.921	19.823	11.908	10.709	13.309	14.704	14.747	23.850	14.079
1950	15.807	17.950	15.280	15.019	15.651	20.822	19.457	11.106	17.397	10.559
1960	17.149	15.475	14.340	13.670	11.061	12.763	9.792	14.796	11.182	7.078
1970	6.793	2.934	4.516	8.856	10.600	7.564	5.025	3.322	9.463	2.049
1980	1.836	0.648								

SPLINE	SIG05-2									
1880	2.387	2.446	2.505	2.564	2.622	2.681	2.740	2.798	2.857	2.916
1890	2.977	3.040	3.106	3.177	3.252	3.332	3.418	3.511	3.612	3.721
1900	3.840	3.970	4.113	4.268	4.437	4.621	4.821	5.036	5.266	5.512
1910	5.773	6.049	6.340	6.645	6.964	7.294	7.636	7.987	8.345	8.709
1920	9.078	9.449	9.823	10.197	10.569	10.939	11.304	11.662	12.012	12.353
1930	12.682	13.000	13.305	13.597	13.874	14.136	14.382	14.611	14.822	15.015
1940	15.187	15.338	15.468	15.575	15.661	15.725	15.766	15.784	15.778	15.746
1950	15.687	15.603	15.492	15.353	15.188	14.995	14.774	14.527	14.253	13.955
1960	13.632	13.286	12.917	12.527	12.118	11.690	11.247	10.788	10.316	9.833
1970	9.341	8.841	8.335	7.823	7.307	6.785	6.260	5.731	5.201	4.668
1980	4.134	3.600								

INDEX	SIG05-2									
1880	0.845	1.302	0.887	0.646	1.111	1.316	1.721	1.334	1.780	2.536
1890	0.362	0.142	0.816	1.258	1.369	0.769	0.901	1.497	1.657	1.714
1900	0.233	0.541	0.530	0.958	1.171	0.109	0.358	0.937	0.857	1.157
1910	0.733	0.600	0.691	0.684	0.630	0.668	0.930	1.374	0.852	1.099
1920	1.059	1.009	0.784	0.999	0.782	1.134	0.998	1.039	1.432	1.231
1930	0.853	0.971	0.855	1.341	1.049	0.985	0.653	0.912	1.145	1.043
1940	1.158	1.299	1.282	0.765	0.684	0.846	0.933	0.934	1.512	0.894
1950	1.008	1.150	0.986	0.978	1.031	1.389	1.317	0.765	1.221	0.757
1960	1.258	1.165	1.110	1.091	0.913	1.092	0.871	1.371	1.084	0.720
1970	0.727	0.332	0.542	1.132	1.451	1.115	0.803	0.580	1.820	0.439
1980	0.444	0.180								

Output file KNOB.IND, continued

RING WIDTHS	SIG06-1	MS=	0.396	1882=	0.00	CM.
1882	0.483	0.268	0.210	0.172	0.313	0.237
1890	0.052	0.059	0.107	0.147	0.196	0.119
1900	0.114	0.182	0.158	0.226	0.159	0.109
1910	0.136	0.141	0.155	0.213	0.117	0.167
1920	0.141	0.126	0.133	0.190	0.163	0.165
1930	0.096	0.000	0.072	0.126	0.147	0.159
1940	0.171	0.135	0.085	0.070	0.054	0.103
1950	0.138	0.112	0.135	0.102	0.098	0.095
1960	0.082	0.062	0.065	0.091	0.051	0.046
1970	0.062	0.041	0.033	0.100	0.066	0.063
1980	0.027	0.022				

AREA	SIG06-1
1882	5.825
1890	1.299
1900	3.980
1910	6.329
1920	8.040
1930	6.402
1940	12.693
1950	11.224
1960	7.164
1970	5.638
1980	2.539

SPLINE	SIG06-1
1882	3.673
1890	4.817
1900	6.402
1910	7.874
1920	8.796
1930	9.110
1940	9.064
1950	8.380
1960	6.852
1970	5.022
1980	3.163

INDEX	SIG06-1
1882	1.586
1890	0.270
1900	0.622
1910	0.804
1920	0.914
1930	0.703
1940	1.400
1950	1.339
1960	1.046
1970	1.123
1980	0.803

RING WIDTHS	SIG06-2	MS=	0.384	1880=	0.00	CM.
1880	0.289	0.300	0.341	0.196	0.189	0.213
1890	0.017	0.055	0.102	0.110	0.207	0.134
1900	0.092	0.157	0.126	0.255	0.178	0.120
1910	0.106	0.105	0.138	0.155	0.082	0.129
1920	0.114	0.120	0.107	0.124	0.138	0.149
1930	0.081	0.000	0.060	0.086	0.121	0.131
1940	0.137	0.115	0.045	0.076	0.046	0.104
1950	0.132	0.102	0.097	0.085	0.092	0.086
1960	0.100	0.087	0.071	0.065	0.044	0.045
1970	0.047	0.045	0.044	0.080	0.055	0.061
1980	0.052	0.032				

AREA	SIG06-2
1880	2.732
1890	0.453
1900	3.257
1910	4.860

Output file KNOB.IND, continued

1920	6.161	6.574	5.938	6.972	7.872	8.634	6.523	5.107	9.504	9.291
1930	5.010	0.000	3.738	5.397	7.672	8.410	5.775	7.800	9.825	12.610
1940	9.378	7.963	3.139	5.330	3.243	7.382	9.540	9.359	8.947	7.255
1950	9.867	7.700	7.383	6.518	7.106	6.691	7.445	5.759	8.422	5.119
1960	8.051	7.055	5.793	5.331	3.624	3.719	3.732	5.079	4.934	4.030
1970	3.961	3.805	3.733	6.818	4.711	5.247	4.491	2.599	6.783	3.144
1980	4.555	2.812								

SPLINE	SIG06-2									
1880	3.867	3.960	4.052	4.144	4.235	4.325	4.415	4.502	4.589	4.675
1890	4.761	4.848	4.937	5.028	5.119	5.212	5.305	5.397	5.490	5.581
1900	5.671	5.759	5.845	5.928	6.009	6.085	6.158	6.227	6.292	6.352
1910	6.409	6.463	6.514	6.562	6.607	6.650	6.691	6.729	6.765	6.799
1920	6.830	6.859	6.887	6.913	6.939	6.962	6.985	7.008	7.030	7.052
1930	7.074	7.097	7.121	7.147	7.173	7.197	7.220	7.241	7.257	7.269
1940	7.277	7.279	7.276	7.268	7.256	7.239	7.215	7.185	7.148	7.104
1950	7.051	6.991	6.923	6.849	6.767	6.680	6.586	6.487	6.383	6.274
1960	6.162	6.046	5.928	5.808	5.687	5.566	5.446	5.326	5.208	5.091
1970	4.975	4.860	4.746	4.633	4.520	4.408	4.295	4.183	4.070	3.957
1980	3.844	3.731								

INDEX	SIG06-2									
1880	0.706	0.856	1.121	0.710	0.724	0.861	2.857	1.423	1.088	1.294
1890	0.095	0.305	0.565	0.613	1.174	0.774	0.894	1.336	1.196	0.980
1900	0.574	0.986	0.799	1.646	1.174	0.800	1.084	1.305	1.533	1.106
1910	0.758	0.756	1.002	1.138	0.607	0.962	1.063	1.284	1.022	1.135
1920	0.902	0.958	0.862	1.008	1.135	1.240	0.934	0.729	1.352	1.318
1930	0.708	0.000	0.525	0.755	1.070	1.168	0.800	1.077	1.354	1.735
1940	1.289	1.094	0.431	0.733	0.447	1.020	1.322	1.302	1.252	1.021
1950	1.399	1.101	1.066	0.952	1.050	1.002	1.130	0.888	1.320	0.816
1960	1.307	1.167	0.977	0.918	0.637	0.668	0.685	0.953	0.947	0.792
1970	0.796	0.783	0.786	1.472	1.042	1.190	1.046	0.621	1.667	0.794
1980	1.185	0.754								

RING WIDTHS	SIG08-1	MS=	0.336	1887=	0.00 CM.					
1887	0.471	0.358	0.661							
1890	0.126	0.148	0.200	0.228	0.235	0.136	0.194	0.270	0.255	0.180
1900	0.146	0.183	0.181	0.231	0.195	0.158	0.159	0.248	0.278	0.183
1910	0.183	0.171	0.171	0.199	0.151	0.142	0.198	0.157	0.193	0.171
1920	0.210	0.207	0.171	0.129	0.122	0.084	0.131	0.114	0.180	0.220
1930	0.134	0.172	0.125	0.174	0.166	0.125	0.127	0.165	0.151	0.220
1940	0.207	0.247	0.116	0.114	0.088	0.249	0.140	0.141	0.237	0.207
1950	0.250	0.184	0.166	0.196	0.118	0.172	0.158	0.091	0.208	0.148
1960	0.139	0.166	0.111	0.161	0.118	0.115	0.096	0.161	0.135	0.063
1970	0.108	0.038	0.088	0.128	0.133	0.047	0.075	0.069	0.099	0.030
1980	0.068	0.041								

AREA	SIG08-1									
1887	9.833	8.406	17.636							
1890	3.673	4.442	6.222	7.399	7.968	4.770	7.005	10.143	10.000	7.305
1900	6.075	7.803	7.925	10.413	9.051	7.509	7.715	12.351	14.304	9.681
1910	9.891	9.433	9.617	11.423	8.833	8.438	11.977	9.672	12.102	10.918
1920	13.659	13.735	11.550	8.834	8.451	5.873	9.248	8.136	13.012	16.180
1930	10.004	13.006	9.569	13.483	13.041	9.934	10.194	13.395	12.408	18.335
1940	17.529	21.269	10.121	10.029	7.797	22.326	12.724	12.939	22.031	19.531
1950	23.947	17.876	16.310	19.480	11.844	17.421	16.167	9.382	21.641	15.564
1960	14.743	17.766	11.976	17.508	12.935	12.691	10.658	18.004	15.222	7.143
1970	12.303	4.346	10.100	14.777	15.464	5.491	8.791	8.119	11.701	3.558
1980	8.086	4.889								

SPLINE	SIG08-1									
1887	7.742	7.791	7.841							
1890	7.891	7.943	7.998	8.057	8.120	8.187	8.258	8.334	8.414	8.498
1900	8.587	8.679	8.775	8.874	8.977	9.081	9.188	9.296	9.405	9.515
1910	9.626	9.737	9.850	9.964	10.080	10.199	10.319	10.443	10.568	10.697
1920	10.829	10.965	11.105	11.250	11.402	11.560	11.726	11.900	12.080	12.266
1930	12.458	12.653	12.853	13.056	13.260	13.466	13.672	13.877	14.078	14.274
1940	14.463	14.643	14.813	14.972	15.119	15.253	15.371	15.471	15.552	15.611
1950	15.647	15.658	15.645	15.607	15.546	15.462	15.355	15.225	15.074	14.901
1960	14.708	14.494	14.261	14.010	13.741	13.457	13.159	12.847	12.523	12.188

Output file KNOB.IND, continued

1970 11.844 11.492 11.133 10.768 10.397 10.021 9.642 9.259 8.874 8.487
1980 8.100 7.712

INDEX SIG08-1
1887 1.270 1.079 2.249
1890 0.466 0.559 0.778 0.918 0.981 0.583 0.848 1.217 1.188 0.860
1900 0.707 0.899 0.903 1.173 1.008 0.827 0.840 1.329 1.521 1.017
1910 1.028 0.969 0.976 1.146 0.876 0.827 1.161 0.926 1.145 1.021
1920 1.261 1.253 1.040 0.785 0.741 0.508 0.789 0.684 1.077 1.319
1930 0.803 1.028 0.744 1.033 0.983 0.738 0.746 0.965 0.881 1.284
1940 1.212 1.452 0.683 0.670 0.516 1.464 0.828 0.836 1.417 1.251
1950 1.530 1.142 1.042 1.248 0.762 1.127 1.053 0.616 1.436 1.044
1960 1.002 1.226 0.840 1.250 0.941 0.943 0.810 1.401 1.215 0.586
1970 1.039 0.378 0.907 1.372 1.487 0.548 0.912 0.877 1.319 0.419
1980 0.998 0.634

RING WIDTHS SIG08-2 MS= 0.286 1890= 0.00 CM.
1890 0.058 0.097 0.179 0.200 0.205 0.122 0.185 0.262 0.258 0.225
1900 0.149 0.200 0.169 0.170 0.220 0.147 0.229 0.177 0.186 0.184
1910 0.139 0.107 0.142 0.125 0.101 0.139 0.173 0.149 0.207 0.199
1920 0.196 0.204 0.154 0.120 0.094 0.099 0.139 0.130 0.196 0.224
1930 0.158 0.119 0.121 0.173 0.154 0.148 0.106 0.184 0.184 0.251
1940 0.153 0.162 0.113 0.109 0.069 0.196 0.211 0.194 0.259 0.195
1950 0.190 0.138 0.177 0.179 0.126 0.181 0.174 0.096 0.140 0.086
1960 0.117 0.136 0.090 0.132 0.120 0.100 0.093 0.140 0.119 0.053
1970 0.099 0.041 0.079 0.116 0.156 0.053 0.080 0.057 0.059 0.020
1980 0.056 0.018

AREA SIG08-2
1890 1.163 1.991 3.830 4.518 4.891 3.036 4.783 7.141 7.454 6.842
1900 4.706 6.536 5.719 5.934 7.948 5.480 8.808 7.034 7.603 7.735
1910 5.985 4.690 6.335 5.681 4.662 6.521 8.286 7.287 10.355 10.207
1920 10.298 10.975 8.458 6.694 5.307 5.649 8.035 7.625 11.697 13.663
1930 9.827 7.505 7.722 11.201 10.129 9.875 7.157 12.591 12.804 17.809
1940 11.050 11.860 8.371 8.150 5.198 14.928 16.341 15.271 20.756 15.905
1950 15.727 11.565 15.009 15.378 10.946 15.898 15.477 8.621 12.676 7.848
1960 10.751 12.605 8.405 12.420 11.386 9.557 8.945 13.568 11.629 5.208
1970 9.776 4.067 7.865 11.620 15.760 5.389 8.168 5.844 6.071 2.063
1980 5.789 1.865

SPLINE SIG08-2
1890 3.617 3.805 3.993 4.180 4.366 4.550 4.732 4.910 5.086 5.258
1900 5.425 5.589 5.750 5.907 6.060 6.211 6.358 6.503 6.647 6.789
1910 6.930 7.072 7.216 7.361 7.508 7.657 7.808 7.962 8.116 8.272
1920 8.430 8.588 8.749 8.912 9.079 9.248 9.422 9.598 9.776 9.956
1930 10.137 10.318 10.500 10.681 10.861 11.039 11.214 11.385 11.550 11.708
1940 11.857 11.998 12.129 12.249 12.358 12.454 12.534 12.596 12.640 12.662
1950 12.663 12.643 12.603 12.541 12.459 12.358 12.239 12.101 11.947 11.779
1960 11.596 11.400 11.191 10.970 10.737 10.493 10.239 9.975 9.701 9.418
1970 9.128 8.830 8.526 8.214 7.897 7.573 7.243 6.910 6.574 6.236
1980 5.897 5.558

INDEX SIG08-2
1890 0.321 0.523 0.959 1.081 1.120 0.667 1.011 1.454 1.466 1.301
1900 0.867 1.169 0.995 1.005 1.312 0.882 1.385 1.082 1.144 1.139
1910 0.864 0.663 0.878 0.772 0.621 0.852 1.061 0.915 1.276 1.234
1920 1.222 1.278 0.967 0.751 0.585 0.611 0.853 0.794 1.196 1.372
1930 0.969 0.727 0.735 1.049 0.933 0.894 0.638 1.106 1.109 1.521
1940 0.932 0.989 0.690 0.665 0.421 1.199 1.304 1.212 1.642 1.256
1950 1.242 0.915 1.191 1.226 0.879 1.286 1.265 0.712 1.061 0.666
1960 0.927 1.106 0.751 1.132 1.060 0.911 0.874 1.360 1.199 0.553
1970 1.071 0.461 0.923 1.415 1.996 0.712 1.128 0.846 0.923 0.331
1980 0.982 0.336

RING WIDTHS SIG09-1 MS= 0.481 1890= 0.00 CM.
1890 0.179 0.181 0.187 0.251 0.256 0.097 0.088 0.116 0.192 0.174
1900 0.127 0.182 0.122 0.201 0.140 0.062 0.150 0.200 0.228 0.137
1910 0.182 0.159 0.191 0.233 0.096 0.171 0.156 0.236 0.183 0.158
1920 0.224 0.172 0.110 0.096 0.147 0.146 0.127 0.130 0.190 0.175
1930 0.084 0.000 0.016 0.082 0.082 0.115 0.112 0.167 0.198 0.188

Output file KNOB.IND, continued

1940	0.183	0.185	0.102	0.120	0.042	0.152	0.142	0.124	0.144	0.106
1950	0.110	0.124	0.084	0.114	0.093	0.126	0.086	0.070	0.106	0.075
1960	0.093	0.078	0.085	0.090	0.046	0.048	0.043	0.111	0.099	0.063
1970	0.067	0.000	0.048	0.038	0.025	0.048	0.032	0.020	0.040	0.016
1980	0.034	0.030								

AREA	SIG09-1									
1890	6.244	6.518	6.950	9.675	10.275	4.001	3.681	4.926	8.340	7.758
1900	5.782	8.463	5.790	9.743	6.936	3.111	7.627	10.389	12.150	7.458
1910	10.089	8.985	11.003	13.733	5.757	10.399	9.647	14.885	11.783	10.342
1920	14.931	11.679	7.567	6.666	10.319	10.383	9.141	9.462	14.020	13.114
1930	6.363	0.000	1.217	6.263	6.305	8.913	8.761	13.209	15.888	15.314
1940	15.120	15.499	8.637	10.245	3.607	13.147	12.413	10.943	12.830	9.527
1950	9.962	11.321	7.724	10.553	8.669	11.832	8.133	6.655	10.135	7.214
1960	8.994	7.586	8.310	8.848	4.542	4.754	4.271	11.078	9.946	6.361
1970	6.793	0.000	4.884	3.877	2.555	4.917	3.286	2.057	4.122	1.652
1980	3.515	3.107								

SPLINE	SIG09-1									
1890	6.357	6.488	6.618	6.748	6.879	7.010	7.142	7.275	7.410	7.547
1900	7.684	7.823	7.962	8.102	8.242	8.381	8.519	8.655	8.788	8.916
1910	9.039	9.156	9.267	9.370	9.466	9.554	9.634	9.706	9.769	9.823
1920	9.870	9.909	9.943	9.971	9.997	10.019	10.040	10.059	10.078	10.097
1930	10.118	10.142	10.170	10.201	10.234	10.268	10.301	10.330	10.353	10.367
1940	10.371	10.365	10.347	10.317	10.277	10.226	10.162	10.087	9.999	9.899
1950	9.786	9.662	9.526	9.379	9.221	9.052	8.873	8.685	8.487	8.281
1960	8.067	7.845	7.616	7.381	7.139	6.891	6.639	6.382	6.120	5.853
1970	5.583	5.310	5.036	4.761	4.485	4.209	3.932	3.656	3.380	3.105
1980	2.829	2.554								

INDEX	SIG09-1									
1890	0.982	1.005	1.050	1.434	1.494	0.571	0.515	0.677	1.125	1.028
1900	0.752	1.082	0.727	1.202	0.842	0.371	0.895	1.200	1.383	0.836
1910	1.116	0.981	1.187	1.466	0.608	1.088	1.001	1.534	1.206	1.053
1920	1.513	1.179	0.761	0.668	1.032	1.036	0.911	0.941	1.391	1.299
1930	0.629	0.000	0.120	0.614	0.616	0.868	0.850	1.279	1.535	1.477
1940	1.458	1.495	0.835	0.993	0.351	1.286	1.221	1.085	1.283	0.962
1950	1.018	1.172	0.811	1.125	0.940	1.307	0.917	0.766	1.194	0.871
1960	1.115	0.967	1.091	1.199	0.636	0.690	0.643	1.736	1.625	1.087
1970	1.217	0.000	0.970	0.814	0.570	1.168	0.836	0.563	1.219	0.532
1980	1.242	1.217								

RING WIDTHS	SIG09-2	MS=	0.277	1890=	0.00	CM.				
1890	0.341	0.287	0.310	0.259	0.255	0.238	0.082	0.135	0.127	0.199
1900	0.154	0.127	0.206	0.152	0.207	0.194	0.118	0.160	0.246	0.208
1910	0.247	0.243	0.230	0.248	0.246	0.118	0.175	0.171	0.223	0.183
1920	0.246	0.249	0.286	0.198	0.205	0.191	0.183	0.226	0.146	0.237
1930	0.232	0.139	0.068	0.111	0.124	0.177	0.165	0.203	0.213	0.243
1940	0.213	0.242	0.228	0.127	0.099	0.255	0.176	0.266	0.293	0.214
1950	0.211	0.190	0.144	0.149	0.148	0.163	0.130	0.098	0.178	0.094
1960	0.135	0.116	0.120	0.110	0.068	0.060	0.068	0.110	0.144	0.120
1970	0.105	0.077	0.014	0.090	0.072	0.103	0.076	0.041	0.059	0.062
1980	0.082	0.064								

AREA	SIG09-2									
1890	7.982	7.284	8.449	7.522	7.818	7.665	2.723	4.576	4.409	7.113
1900	5.675	4.772	7.989	6.065	8.494	8.205	5.106	7.063	11.174	9.744
1910	11.924	12.105	11.799	13.095	13.372	6.549	9.873	9.834	13.100	10.984
1920	15.096	15.458	18.477	13.093	13.815	13.109	12.775	16.067	10.550	17.411
1930	17.386	10.579	5.219	8.582	9.679	13.983	13.213	16.490	17.581	20.405
1940	18.191	21.014	20.135	11.357	8.923	23.268	16.298	25.001	28.054	20.831
1950	20.820	18.987	14.542	15.184	15.220	16.922	13.615	10.334	18.925	10.074
1960	14.565	12.607	13.131	12.116	7.528	6.666	7.583	12.327	16.253	13.643
1970	12.012	8.853	1.614	10.403	8.359	12.014	8.908	4.821	6.955	7.333
1980	9.735	7.627								

SPLINE	SIG09-2									
1890	5.523	5.705	5.887	6.070	6.255	6.442	6.633	6.829	7.030	7.238
1900	7.453	7.674	7.903	8.138	8.379	8.626	8.878	9.134	9.394	9.656
1910	9.919	10.182	10.444	10.705	10.964	11.221	11.476	11.729	11.978	12.224

Output file KNOB.IND, continued

1920	12.465	12.701	12.931	13.155	13.373	13.586	13.794	13.998	14.198	14.395
1930	14.589	14.780	14.969	15.156	15.341	15.522	15.697	15.863	16.018	16.158
1940	16.282	16.387	16.472	16.535	16.576	16.592	16.583	16.545	16.479	16.383
1950	16.257	16.105	15.927	15.725	15.502	15.260	15.000	14.726	14.439	14.140
1960	13.831	13.515	13.193	12.866	12.536	12.205	11.872	11.539	11.205	10.870
1970	10.534	10.199	9.864	9.530	9.198	8.866	8.536	8.207	7.878	7.551
1980	7.224	6.897								

INDEX	SIG09-2									
1890	1.445	1.277	1.435	1.239	1.250	1.190	0.411	0.670	0.627	0.983
1900	0.761	0.624	1.011	0.745	1.014	0.951	0.575	0.773	1.189	1.009
1910	1.202	1.189	1.130	1.223	1.220	0.584	0.860	0.838	1.094	0.899
1920	1.211	1.234	1.429	0.995	1.033	0.965	0.926	1.148	0.743	1.210
1930	1.192	0.716	0.349	0.566	0.631	0.901	0.842	1.040	1.098	1.263
1940	1.117	1.282	1.222	0.687	0.538	1.402	0.983	1.511	1.702	1.272
1950	1.281	1.179	0.913	0.966	0.982	1.109	0.908	0.702	1.311	0.712
1960	1.053	0.933	0.995	0.942	0.600	0.546	0.639	1.068	1.450	1.255
1970	1.140	0.868	0.164	1.092	0.909	1.355	1.044	0.587	0.883	0.971
1980	1.348	1.106								

RING	WIDTHS	SIG11-1 MS= 0.391 1863= 0.00 CM.									
1863	0.065	0.105	0.084	0.156	0.166	0.207	0.224				
1870	0.205	0.015	0.060	0.091	0.088	0.138	0.222	0.220	0.323	0.231	
1880	0.314	0.239	0.413	0.250	0.289	0.190	0.276	0.200	0.234	0.258	
1890	0.214	0.204	0.191	0.210	0.224	0.141	0.136	0.183	0.191	0.133	
1900	0.098	0.114	0.093	0.195	0.144	0.144	0.155	0.173	0.184	0.225	
1910	0.146	0.124	0.156	0.170	0.106	0.136	0.142	0.179	0.128	0.128	
1920	0.142	0.167	0.149	0.078	0.082	0.090	0.109	0.106	0.180	0.158	
1930	0.086	0.000	0.031	0.080	0.048	0.084	0.043	0.082	0.130	0.155	
1940	0.142	0.148	0.100	0.109	0.076	0.164	0.109	0.151	0.179	0.186	
1950	0.160	0.163	0.133	0.117	0.133	0.136	0.136	0.102	0.174	0.105	
1960	0.110	0.094	0.100	0.108	0.063	0.067	0.031	0.100	0.060	0.050	
1970	0.056	0.026	0.048	0.070	0.096	0.111	0.109	0.028	0.054	0.042	
1980	0.025	0.000									

AREA	SIG11-1										
1863	0.830	1.397	1.167	2.286	2.600	3.485	4.075				
1870	4.005	0.303	1.228	1.905	1.892	3.065	5.182	5.441	8.539	6.509	
1880	9.385	7.559	13.908	8.939	10.823	7.402	11.156	8.383	10.127	11.565	
1890	9.910	9.715	9.431	10.528	11.535	7.423	7.278	9.976	10.637	7.542	
1900	5.629	6.623	5.464	11.633	8.744	8.874	9.698	11.002	11.908	14.850	
1910	9.806	8.434	10.748	11.886	7.503	9.730	10.284	13.144	9.522	9.625	
1920	10.798	12.862	11.623	6.140	6.496	7.179	8.763	8.593	14.754	13.118	
1930	7.206	0.000	2.609	6.761	4.076	7.167	3.686	7.062	11.282	13.590	
1940	12.583	13.249	9.030	9.914	6.957	15.136	10.153	14.189	17.006	17.884	
1950	15.558	16.015	13.191	11.696	13.400	13.511	13.931	10.525	18.105	11.017	
1960	11.616	9.987	10.685	11.611	6.807	7.266	3.372	10.917	6.580	5.501	
1970	6.180	2.876	5.320	7.785	10.726	12.475	12.325	3.178	6.143	4.791	
1980	2.857	0.000									

SPLINE	SIG11-1										
1863	1.258	1.599	1.940	2.280	2.620	2.960	3.299				
1870	3.638	3.975	4.312	4.648	4.981	5.312	5.638	5.958	6.269	6.571	
1880	6.861	7.138	7.401	7.648	7.879	8.093	8.291	8.473	8.638	8.788	
1890	8.922	9.041	9.147	9.240	9.322	9.393	9.455	9.510	9.557	9.599	
1900	9.636	9.669	9.699	9.725	9.748	9.766	9.781	9.790	9.795	9.793	
1910	9.787	9.775	9.758	9.738	9.714	9.687	9.657	9.626	9.593	9.560	
1920	9.527	9.497	9.469	9.445	9.428	9.418	9.417	9.425	9.443	9.472	
1930	9.513	9.568	9.638	9.723	9.823	9.936	10.062	10.198	10.341	10.489	
1940	10.638	10.787	10.931	11.070	11.202	11.322	11.430	11.521	11.594	11.646	
1950	11.676	11.682	11.664	11.623	11.558	11.469	11.357	11.223	11.068	10.893	
1960	10.699	10.488	10.263	10.024	9.774	9.516	9.249	8.977	8.700	8.419	
1970	8.134	7.846	7.556	7.261	6.963	6.661	6.353	6.041	5.726	5.409	
1980	5.090	4.770									

INDEX	SIG11-1										
1863	0.660	0.874	0.602	1.002	0.992	1.177	1.235				
1870	1.101	0.076	0.285	0.410	0.380	0.577	0.919	0.913	1.362	0.991	
1880	1.368	1.059	1.879	1.169	1.374	0.915	1.346	0.989	1.172	1.316	
1890	1.111	1.074	1.031	1.139	1.237	0.790	0.770	1.049	1.113	0.786	

Output file KNOB.IND, continued

1900	0.584	0.685	0.563	1.196	0.897	0.909	0.991	1.124	1.216	1.516
1910	1.002	0.863	1.101	1.221	0.772	1.005	1.065	1.365	0.993	1.007
1920	1.133	1.354	1.228	0.650	0.689	0.762	0.931	0.912	1.562	1.385
1930	0.757	0.000	0.271	0.695	0.415	0.721	0.366	0.692	1.091	1.296
1940	1.183	1.228	0.826	0.896	0.621	1.337	0.888	1.232	1.467	1.536
1950	1.333	1.371	1.131	1.006	1.159	1.178	1.227	0.938	1.636	1.011
1960	1.086	0.952	1.041	1.158	0.696	0.764	0.365	1.216	0.756	0.653
1970	0.760	0.367	0.704	1.072	1.540	1.873	1.940	0.526	1.073	0.886
1980	0.561	0.000								

RING WIDTHS SIG11-2 MS= 0.389 1870= 0.00 CM.

1870	0.007	0.040	0.058	0.144	0.115	0.159	0.193	0.234	0.386	0.261
1880	0.246	0.219	0.375	0.216	0.214	0.133	0.263	0.240	0.262	0.244
1890	0.157	0.122	0.146	0.190	0.213	0.124	0.076	0.174	0.149	0.112
1900	0.041	0.062	0.080	0.148	0.138	0.097	0.095	0.192	0.251	0.200
1910	0.107	0.120	0.136	0.158	0.064	0.107	0.113	0.114	0.099	0.110
1920	0.107	0.083	0.068	0.083	0.070	0.055	0.081	0.079	0.108	0.104
1930	0.052	0.000	0.024	0.047	0.052	0.071	0.037	0.051	0.065	0.110
1940	0.086	0.113	0.089	0.067	0.042	0.106	0.066	0.078	0.120	0.130
1950	0.115	0.113	0.080	0.105	0.073	0.080	0.103	0.069	0.104	0.073
1960	0.049	0.067	0.052	0.083	0.049	0.062	0.034	0.065	0.041	0.029
1970	0.063	0.014	0.026	0.062	0.066	0.075	0.064	0.040	0.039	0.046
1980	0.045	0.016								

AREA SIG11-2

1870	0.213	1.226	1.795	4.548	3.726	5.288	6.633	8.356	14.535	10.358
1880	10.155	9.360	16.728	10.036	10.232	6.504	13.189	12.415	13.966	13.395
1890	8.816	6.958	8.450	11.197	12.822	7.596	4.703	10.904	9.489	7.224
1900	2.664	4.049	5.260	9.838	9.297	6.606	6.527	13.365	17.822	14.484
1910	7.852	8.892	10.187	11.980	4.897	8.245	8.786	8.945	7.834	8.777
1920	8.610	6.729	5.545	8.807	5.775	4.559	6.749	6.622	9.116	8.848
1930	4.449	0.000	2.059	4.043	4.490	6.157	3.221	4.454	5.701	9.708
1940	7.643	10.113	8.021	6.071	3.820	9.691	6.070	7.209	11.165	12.197
1950	10.879	10.770	7.674	10.133	7.085	7.803	10.106	6.807	10.317	7.282
1960	4.907	6.734	5.246	8.408	4.984	6.328	3.480	6.674	4.223	2.994
1970	6.522	1.453	2.701	6.458	6.901	7.876	6.748	4.231	4.135	4.889
1980	4.796	1.708								

SPLINE SIG11-2

1870	5.435	5.741	6.047	6.351	6.651	6.946	7.234	7.511	7.775	8.025
1880	8.258	8.474	8.670	8.846	9.002	9.139	9.255	9.352	9.430	9.489
1890	9.531	9.558	9.571	9.572	9.561	9.542	9.515	9.481	9.443	9.401
1900	9.356	9.310	9.262	9.212	9.159	9.103	9.042	8.976	8.902	8.820
1910	8.730	8.633	8.530	8.421	8.307	8.190	8.070	7.949	7.828	7.707
1920	7.588	7.472	7.361	7.256	7.158	7.067	6.986	6.915	6.854	6.803
1930	6.764	6.738	6.724	6.724	6.737	6.761	6.796	6.840	6.892	6.949
1940	7.010	7.073	7.135	7.197	7.255	7.310	7.358	7.400	7.432	7.454
1950	7.465	7.462	7.447	7.419	7.379	7.327	7.263	7.188	7.103	7.009
1960	6.907	6.797	6.682	6.562	6.437	6.310	6.180	6.049	5.916	5.784
1970	5.652	5.519	5.388	5.256	5.124	4.990	4.856	4.720	4.583	4.446
1980	4.308	4.169								

INDEX SIG11-2

1870	0.039	0.213	0.297	0.716	0.560	0.761	0.917	1.112	1.869	1.291
1880	1.230	1.105	1.929	1.135	1.137	0.712	1.425	1.328	1.481	1.412
1890	0.925	0.728	0.883	1.170	1.341	0.796	0.494	1.150	1.005	0.768
1900	0.285	0.435	0.568	1.068	1.015	0.722	0.722	1.489	2.002	1.642
1910	0.899	1.030	1.194	1.423	0.590	1.007	1.089	1.125	1.001	1.139
1920	1.135	0.901	0.753	0.938	0.807	0.645	0.966	0.958	1.330	1.301
1930	0.658	0.000	0.306	0.601	0.666	0.911	0.474	0.651	0.827	1.397
1940	1.090	1.430	1.124	0.844	0.527	1.326	0.825	0.974	1.502	1.636
1950	1.457	1.443	1.030	1.366	0.960	1.065	1.391	0.947	1.452	1.039
1960	0.710	0.991	0.785	1.281	0.774	1.003	0.563	1.103	0.714	0.518
1970	1.154	0.263	0.501	1.229	1.347	1.578	1.390	0.896	0.902	1.100
1980	1.113	0.410								

RING WIDTHS SIG12-1 MS= 0.445 1882= 0.00 CM.

1882	0.302	0.314	0.300	0.308	0.206	0.301	0.323	0.322		
1890	0.061	0.068	0.121	0.141	0.185	0.038	0.037	0.087	0.189	0.154
1900	0.028	0.130	0.116	0.194	0.194	0.288	0.274	0.201	0.230	0.209

Output file KNOB.IND, continued

1910	0.258	0.207	0.352	0.227	0.167	0.248	0.141	0.213	0.230	0.191
1920	0.264	0.211	0.177	0.046	0.050	0.077	0.126	0.106	0.171	0.174
1930	0.093	0.021	0.051	0.072	0.132	0.143	0.144	0.163	0.180	0.187
1940	0.137	0.117	0.124	0.088	0.053	0.076	0.040	0.049	0.166	0.152
1950	0.127	0.158	0.120	0.117	0.122	0.113	0.118	0.080	0.178	0.145
1960	0.124	0.127	0.073	0.174	0.077	0.047	0.056	0.084	0.135	0.093
1970	0.080	0.030	0.039	0.074	0.107	0.127	0.079	0.074	0.114	0.049
1980	0.125	0.116								

AREA	SIG12-1									
1882	1.431	2.095	2.581	3.238	2.498	4.130	5.065	5.701		
1890	1.153	1.313	2.409	2.923	4.025	0.853	0.840	2.008	4.526	3.854
1900	0.717	3.392	3.117	5.401	5.638	8.805	8.861	6.800	8.093	7.642
1910	9.812	8.175	14.520	9.776	7.399	11.311	6.603	10.212	11.347	9.676
1920	13.751	11.305	9.699	2.553	2.790	4.327	7.162	6.102	9.993	10.357
1930	5.613	1.275	3.108	4.416	8.180	8.985	9.178	10.546	11.840	12.516
1940	9.309	8.043	8.619	6.175	3.743	5.397	2.855	3.511	12.008	11.147
1950	9.425	11.867	9.118	8.977	9.452	8.838	9.315	6.365	14.307	11.801
1960	10.197	10.544	6.107	14.690	6.562	4.023	4.812	7.255	9.131	8.145
1970	7.050	2.654	3.459	6.589	9.589	11.474	7.189	6.769	10.496	4.536
1980	11.641	10.890								

SPLINE	SIG12-1									
1882	1.650	1.830	2.010	2.190	2.370	2.550	2.731	2.912		
1890	3.095	3.281	3.470	3.663	3.861	4.063	4.270	4.482	4.698	4.919
1900	5.142	5.367	5.594	5.819	6.041	6.258	6.469	6.671	6.862	7.043
1910	7.210	7.363	7.501	7.624	7.730	7.821	7.897	7.959	8.006	8.040
1920	8.062	8.073	8.075	8.072	8.064	8.055	8.046	8.037	8.029	8.024
1930	8.021	8.023	8.028	8.039	8.053	8.070	8.090	8.110	8.131	8.151
1940	8.171	8.192	8.213	8.235	8.260	8.287	8.316	8.346	8.377	8.407
1950	8.435	8.459	8.479	8.494	8.505	8.509	8.509	8.502	8.490	8.472
1960	8.448	8.419	8.387	8.351	8.314	8.277	8.242	8.209	8.179	8.153
1970	8.131	8.114	8.103	8.096	8.094	8.096	8.100	8.107	8.115	8.125
1980	8.137	8.148								

INDEX	SIG12-1									
1882	0.867	1.145	1.284	1.477	1.054	1.620	1.855	1.958		
1890	0.373	0.400	0.694	0.798	1.042	0.210	0.197	0.448	0.963	0.784
1900	0.139	0.632	0.557	0.928	0.933	1.407	1.370	1.019	1.179	1.089
1910	1.361	1.110	1.936	1.282	0.957	1.446	0.836	1.283	1.417	1.203
1920	1.706	1.400	1.201	0.316	0.346	0.537	0.890	0.759	1.245	1.291
1930	0.700	0.159	0.387	0.549	1.016	1.113	1.135	1.300	1.456	1.536
1940	1.139	0.982	1.049	0.750	0.453	0.651	0.343	0.421	1.433	1.326
1950	1.117	1.403	1.075	1.057	1.111	1.039	1.095	0.749	1.685	1.393
1960	1.207	1.252	0.728	1.759	0.789	0.486	0.584	0.884	1.116	0.999
1970	0.867	0.327	0.427	0.814	1.185	1.417	0.888	0.835	1.293	0.558
1980	1.431	1.337								

RING WIDTHS	SIG12-2	MS=	0.390	1900=	0.00	CM.				
1900	0.028	0.091	0.050	0.108	0.099	0.179	0.226	0.232	0.311	0.227
1910	0.251	0.188	0.261	0.269	0.150	0.210	0.143	0.206	0.180	0.167
1920	0.189	0.166	0.161	0.025	0.038	0.065	0.130	0.105	0.156	0.152
1930	0.058	0.022	0.052	0.099	0.144	0.125	0.106	0.143	0.132	0.150
1940	0.112	0.106	0.136	0.083	0.045	0.110	0.107	0.130	0.156	0.141
1950	0.123	0.103	0.134	0.135	0.129	0.130	0.087	0.062	0.141	0.082
1960	0.112	0.096	0.082	0.125	0.060	0.055	0.053	0.081	0.094	0.067
1970	0.074	0.044	0.059	0.085	0.080	0.077	0.035	0.035	0.080	0.020
1980	0.046	0.055								

AREA	SIG12-2									
1900	0.467	1.551	0.874	1.942	1.845	3.492	4.696	5.154	7.440	5.814
1910	6.806	5.357	7.805	8.492	4.933	7.144	5.023	7.462	6.738	6.434
1920	7.493	6.766	6.728	1.059	1.618	2.788	5.656	4.646	7.030	6.997
1930	2.708	1.033	2.453	4.717	6.971	6.157	5.298	7.259	6.815	7.877
1940	5.974	5.726	7.450	4.604	2.514	6.200	6.103	7.512	9.155	8.406
1950	7.435	6.299	8.295	8.471	8.201	8.371	5.661	4.063	9.331	5.484
1960	7.559	6.542	5.633	8.669	4.196	3.866	3.744	5.755	6.731	4.831
1970	5.369	3.209	4.322	6.264	5.937	5.753	2.627	2.635	6.051	1.519
1980	3.504	4.207								

Output file KNOB.IND, continued

SPLINE	SIG12-2									
1900	3.174	3.337	3.501	3.663	3.824	3.981	4.135	4.283	4.425	4.559
1910	4.684	4.799	4.905	5.001	5.086	5.162	5.228	5.286	5.335	5.377
1920	5.413	5.445	5.472	5.497	5.522	5.548	5.575	5.605	5.636	5.671
1930	5.708	5.750	5.795	5.844	5.895	5.949	6.005	6.061	6.117	6.172
1940	6.226	6.279	6.329	6.377	6.422	6.464	6.502	6.535	6.561	6.579
1950	6.590	6.593	6.587	6.572	6.548	6.514	6.472	6.422	6.363	6.297
1960	6.223	6.143	6.056	5.963	5.865	5.762	5.656	5.546	5.433	5.317
1970	5.199	5.078	4.955	4.831	4.705	4.577	4.449	4.319	4.190	4.060
1980	3.930	3.800								

INDEX	SIG12-2									
1900	0.147	0.465	0.250	0.530	0.482	0.877	1.136	1.203	1.681	1.275
1910	1.453	1.116	1.591	1.698	0.970	1.384	0.961	1.412	1.263	1.196
1920	1.384	1.243	1.229	0.193	0.293	0.503	1.014	0.829	1.247	1.234
1930	0.474	0.180	0.423	0.807	1.182	1.035	0.882	1.198	1.114	1.276
1940	0.959	0.912	1.177	0.722	0.391	0.959	0.939	1.150	1.395	1.278
1950	1.128	0.955	1.259	1.289	1.253	1.285	0.875	0.633	1.466	0.871
1960	1.215	1.065	0.930	1.454	0.715	0.671	0.662	1.038	1.239	0.909
1970	1.033	0.632	0.872	1.297	1.262	1.257	0.591	0.610	1.444	0.374
1980	0.892	1.107								

RING WIDTHS	SIG13-1 MS= 0.243 1870= 0.00 CM.									
1870	0.024	0.070	0.138	0.192	0.150	0.188	0.257	0.347	0.459	0.354
1880	0.393	0.326	0.519	0.333	0.391	0.351	0.494	0.388	0.402	0.346
1890	0.212	0.227	0.252	0.295	0.289	0.186	0.153	0.224	0.201	0.164
1900	0.094	0.139	0.073	0.099	0.091	0.095	0.088	0.140	0.150	0.123
1910	0.133	0.113	0.142	0.153	0.082	0.109	0.135	0.143	0.119	0.105
1920	0.147	0.172	0.176	0.156	0.135	0.139	0.135	0.116	0.153	0.148
1930	0.089	0.059	0.077	0.125	0.116	0.127	0.093	0.140	0.211	0.184
1940	0.158	0.165	0.103	0.103	0.072	0.151	0.115	0.145	0.160	0.127
1950	0.165	0.129	0.108	0.117	0.122	0.108	0.101	0.088	0.182	0.113
1960	0.083	0.064	0.055	0.063	0.049	0.035	0.030	0.056	0.027	0.029
1970	0.032	0.029	0.023	0.039	0.031	0.015	0.019	0.019	0.020	0.017
1980	0.009	0.000								

AREA	SIG13-1									
1870	0.586	1.730	3.501	5.070	4.122	5.366	7.695	11.048	15.777	13.072
1880	15.434	13.539	22.933	15.605	19.213	18.066	26.737	22.075	23.869	21.357
1890	13.458	14.723	16.723	20.084	20.206	13.282	11.088	16.499	15.074	12.487
1900	7.233	10.798	5.719	7.810	7.233	7.607	7.097	11.391	12.341	10.225
1910	11.163	9.572	12.142	13.224	7.148	9.567	11.953	12.786	10.738	9.549
1920	13.484	15.950	16.514	14.800	12.931	13.434	13.163	11.402	15.168	14.813
1930	8.974	5.976	7.333	12.795	11.961	13.192	9.725	14.742	22.451	19.806
1940	17.177	18.106	11.389	11.456	8.048	16.983	13.030	16.548	18.413	14.730
1950	19.289	15.199	12.806	13.955	14.643	13.041	12.262	10.736	22.358	13.987
1960	10.324	7.991	6.887	7.913	6.172	4.417	3.793	7.095	3.428	3.687
1970	4.074	3.698	2.936	4.987	3.971	1.923	2.438	2.441	2.572	2.188
1980	1.159	0.000								

SPLINE	SIG13-1									
1870	7.650	8.187	8.722	9.256	9.784	10.305	10.815	11.310	11.785	12.237
1880	12.661	13.053	13.410	13.729	14.007	14.242	14.433	14.578	14.679	14.736
1890	14.752	14.730	14.673	14.585	14.468	14.327	14.167	13.991	13.805	13.613
1900	13.419	13.228	13.044	12.870	12.708	12.561	12.431	12.317	12.221	12.142
1910	12.079	12.034	12.005	11.993	11.995	12.013	12.044	12.088	12.144	12.209
1920	12.283	12.363	12.449	12.539	12.633	12.729	12.828	12.929	13.033	13.138
1930	13.244	13.352	13.461	13.569	13.674	13.775	13.869	13.954	14.026	14.084
1940	14.124	14.147	14.151	14.136	14.101	14.045	13.967	13.866	13.739	13.587
1950	13.408	13.202	12.969	12.710	12.425	12.116	11.782	11.426	11.047	10.648
1960	10.231	9.798	9.352	8.896	8.431	7.961	7.487	7.011	6.533	6.054
1970	5.576	5.099	4.622	4.147	3.672	3.198	2.724	2.251	1.778	1.305
1980	0.833	0.360								

INDEX	SIG13-1									
1870	0.077	0.211	0.401	0.548	0.421	0.521	0.712	0.977	1.339	1.068
1880	1.219	1.037	1.710	1.137	1.372	1.268	1.853	1.514	1.626	1.449
1890	0.912	1.000	1.140	1.377	1.397	0.927	0.783	1.179	1.092	0.917
1900	0.539	0.816	0.438	0.607	0.569	0.606	0.571	0.925	1.010	0.842
1910	0.924	0.795	1.011	1.103	0.596	0.796	0.992	1.058	0.884	0.782

Output file KNOB.IND, continued

1920	1.098	1.290	1.326	1.180	1.024	1.055	1.026	0.882	1.164	1.127
1930	0.678	0.448	0.582	0.943	0.875	0.958	0.701	1.056	1.601	1.406
1940	1.216	1.280	0.805	0.810	0.571	1.209	0.933	1.193	1.340	1.084
1950	1.439	1.151	0.987	1.098	1.179	1.076	1.041	0.940	2.024	1.314
1960	1.009	0.816	0.736	0.890	0.732	0.555	0.507	1.012	0.525	0.609
1970	0.731	0.725	0.635	1.203	1.081	0.602	0.895	1.084	1.446	1.676
1980	1.392	0.000								

RING WIDTHS	SIG13-2	MS=	0.283	1870=	0.00	CM.				
1870	0.035	0.064	0.111	0.208	0.208	0.198	0.242	0.290	0.467	0.404
1880	0.384	0.359	0.570	0.279	0.331	0.242	0.435	0.367	0.446	0.428
1890	0.258	0.198	0.217	0.270	0.330	0.173	0.197	0.162	0.149	0.103
1900	0.042	0.049	0.062	0.111	0.093	0.089	0.092	0.144	0.170	0.131
1910	0.096	0.070	0.113	0.121	0.069	0.083	0.102	0.080	0.064	0.096
1920	0.122	0.138	0.111	0.129	0.109	0.090	0.102	0.087	0.133	0.142
1930	0.087	0.066	0.051	0.086	0.091	0.111	0.057	0.090	0.126	0.120
1940	0.108	0.073	0.052	0.044	0.039	0.060	0.046	0.071	0.087	0.084
1950	0.088	0.102	0.085	0.078	0.062	0.102	0.133	0.101	0.119	0.076
1960	0.062	0.076	0.096	0.074	0.052	0.034	0.036	0.046	0.043	0.030
1970	0.032	0.024	0.026	0.031	0.036	0.015	0.005	0.023	0.019	0.016
1980	0.000	0.000								

AREA	SIG13-2									
1870	0.942	1.742	3.082	5.983	6.255	6.207	7.920	9.976	17.176	15.964
1880	16.124	15.913	26.929	13.925	17.155	12.978	24.253	21.386	27.129	27.209
1890	16.958	13.298	14.857	18.899	23.720	12.709	14.701	12.272	11.432	7.984
1900	3.275	3.835	4.874	8.786	7.421	7.152	7.446	11.761	14.052	10.953
1910	8.095	5.939	9.652	10.424	5.986	7.240	8.956	7.070	5.685	8.576
1920	10.982	12.535	10.170	11.916	10.150	8.437	9.624	8.260	12.719	13.703
1930	8.458	6.448	5.001	8.471	9.014	11.065	5.712	9.061	12.771	12.255
1940	11.107	7.549	5.398	4.581	4.070	6.281	4.830	7.482	9.211	8.939
1950	9.412	10.970	9.192	8.475	6.763	11.180	14.675	11.219	13.300	8.541
1960	6.994	8.607	10.924	8.460	5.965	3.910	4.148	5.311	4.977	3.479
1970	3.717	2.792	3.029	3.617	4.208	1.756	0.586	2.696	2.229	1.879
1980	0.000	0.000								

SPLINE	SIG13-2									
1870	8.623	9.117	9.609	10.099	10.584	11.060	11.524	11.972	12.399	12.800
1880	13.171	13.508	13.808	14.068	14.285	14.457	14.585	14.665	14.698	14.685
1890	14.626	14.526	14.388	14.216	14.012	13.783	13.532	13.266	12.988	12.705
1900	12.422	12.143	11.871	11.609	11.358	11.120	10.895	10.684	10.487	10.303
1910	10.132	9.976	9.835	9.708	9.595	9.496	9.410	9.336	9.274	9.222
1920	9.178	9.142	9.110	9.083	9.059	9.038	9.019	9.001	8.984	8.968
1930	8.952	8.937	8.923	8.910	8.897	8.885	8.872	8.859	8.846	8.830
1940	8.814	8.796	8.777	8.759	8.740	8.720	8.699	8.676	8.647	8.612
1950	8.569	8.516	8.451	8.373	8.281	8.173	8.047	7.904	7.741	7.561
1960	7.362	7.146	6.915	6.668	6.407	6.135	5.852	5.560	5.260	4.954
1970	4.642	4.325	4.004	3.681	3.354	3.025	2.695	2.364	2.031	1.698
1980	1.365	1.031								

INDEX	SIG13-2									
1870	0.109	0.191	0.321	0.592	0.591	0.561	0.687	0.833	1.385	1.247
1880	1.224	1.178	1.950	0.990	1.201	0.898	1.663	1.458	1.846	1.853
1890	1.159	0.915	1.033	1.329	1.693	0.922	1.086	0.925	0.880	0.628
1900	0.264	0.316	0.411	0.757	0.653	0.643	0.683	1.101	1.340	1.063
1910	0.799	0.595	0.981	1.074	0.624	0.762	0.952	0.757	0.613	0.930
1920	1.197	1.371	1.116	1.312	1.120	0.933	1.067	0.918	1.416	1.528
1930	0.945	0.721	0.561	0.951	1.013	1.245	0.644	1.023	1.444	1.388
1940	1.260	0.858	0.615	0.523	0.466	0.720	0.555	0.862	1.065	1.038
1950	1.098	1.288	1.088	1.012	0.817	1.368	1.824	1.419	1.718	1.130
1960	0.950	1.204	1.580	1.260	0.931	0.637	0.709	0.955	0.946	0.702
1970	0.801	0.646	0.756	0.983	1.255	0.580	0.217	1.141	1.098	1.107
1980	0.000	0.000								

RING WIDTHS	SIG14-1	MS=	0.261	1880=	0.00	CM.				
1880	0.560	0.379	0.312	0.295	0.179	0.284	0.390	0.340	0.311	0.346
1890	0.192	0.103	0.159	0.183	0.215	0.202	0.214	0.352	0.285	0.271
1900	0.046	0.084	0.086	0.159	0.192	0.172	0.207	0.248	0.235	0.213
1910	0.143	0.129	0.148	0.196	0.134	0.154	0.155	0.172	0.223	0.265
1920	0.216	0.201	0.158	0.141	0.121	0.161	0.148	0.170	0.260	0.229

Output file KNOB.IND, continued

1930	0.134	0.150	0.110	0.166	0.196	0.171	0.083	0.129	0.174	0.228
1940	0.176	0.161	0.079	0.069	0.073	0.120	0.135	0.147	0.200	0.162
1950	0.172	0.142	0.125	0.123	0.154	0.142	0.111	0.079	0.107	0.068
1960	0.080	0.073	0.088	0.104	0.074	0.075	0.060	0.067	0.070	0.040
1970	0.057	0.035	0.057	0.067	0.067	0.073	0.067	0.048	0.063	0.048
1980	0.041	0.018								

AREA	SIG14-1									
1880	4.518	4.176	4.115	4.453	2.969	5.123	7.861	7.633	7.618	9.189
1890	5.424	3.005	4.770	5.686	6.950	6.794	7.477	12.925	11.035	10.967
1900	1.907	3.517	3.647	6.865	8.501	7.812	9.649	11.914	11.646	10.856
1910	7.448	6.829	7.964	10.758	7.494	8.752	8.959	10.119	13.396	16.325
1920	13.633	12.949	10.357	9.375	8.145	10.980	10.237	11.929	18.596	16.730
1930	9.943	11.264	8.350	12.745	15.271	13.520	6.629	10.388	14.178	18.865
1940	14.786	13.696	6.780	5.954	6.332	10.481	11.899	13.087	18.024	14.784
1950	15.877	13.248	11.766	11.674	14.750	13.733	10.823	7.750	10.559	6.748
1960	7.976	7.313	8.861	10.534	7.537	7.674	6.165	6.910	7.250	4.157
1970	5.941	3.658	5.974	7.048	7.076	7.742	7.135	5.129	6.754	5.162
1980	4.421	1.944								

SPLINE	SIG14-1									
1880	4.488	4.659	4.830	5.001	5.172	5.342	5.511	5.679	5.846	6.010
1890	6.174	6.337	6.500	6.662	6.825	6.987	7.148	7.309	7.469	7.629
1900	7.789	7.951	8.116	8.283	8.452	8.622	8.794	8.965	9.137	9.307
1910	9.478	9.648	9.817	9.986	10.154	10.319	10.483	10.642	10.798	10.947
1920	11.089	11.225	11.354	11.476	11.591	11.699	11.800	11.893	11.978	12.052
1930	12.118	12.174	12.222	12.261	12.291	12.312	12.324	12.328	12.322	12.306
1940	12.281	12.246	12.202	12.149	12.088	12.018	11.939	11.848	11.744	11.627
1950	11.495	11.348	11.187	11.012	10.824	10.624	10.413	10.193	9.964	9.729
1960	9.489	9.245	8.998	8.748	8.496	8.243	7.991	7.738	7.486	7.236
1970	6.987	6.739	6.493	6.248	6.002	5.756	5.510	5.263	5.015	4.766
1980	4.516	4.266								

INDEX	SIG14-1									
1880	1.007	0.896	0.852	0.890	0.574	0.959	1.426	1.344	1.303	1.529
1890	0.878	0.474	0.734	0.854	1.018	0.972	1.046	1.768	1.477	1.437
1900	0.245	0.442	0.449	0.829	1.006	0.906	1.097	1.329	1.275	1.166
1910	0.786	0.708	0.811	1.077	0.738	0.848	0.855	0.951	1.241	1.491
1920	1.229	1.154	0.912	0.817	0.703	0.939	0.868	1.003	1.553	1.388
1930	0.820	0.925	0.683	1.039	1.242	1.098	0.538	0.843	1.151	1.533
1940	1.204	1.118	0.556	0.490	0.524	0.872	0.997	1.105	1.535	1.272
1950	1.381	1.167	1.052	1.060	1.363	1.293	1.039	0.760	1.060	0.694
1960	0.841	0.791	0.985	1.204	0.887	0.931	0.771	0.893	0.968	0.574
1970	0.850	0.543	0.920	1.128	1.179	1.345	1.295	0.975	1.347	1.083
1980	0.979	0.456								

RING WIDTHS	SIG14-2	MS=	0.355	1880=	0.00	CM.				
1880	0.507	0.378	0.321	0.296	0.269	0.351	0.897	0.441	0.295	0.436
1890	0.190	0.151	0.160	0.271	0.289	0.219	0.252	0.313	0.289	0.289
1900	0.024	0.045	0.086	0.107	0.158	0.145	0.220	0.223	0.295	0.225
1910	0.116	0.076	0.110	0.189	0.152	0.158	0.201	0.229	0.210	0.182
1920	0.279	0.349	0.224	0.189	0.171	0.159	0.182	0.182	0.295	0.276
1930	0.174	0.126	0.096	0.213	0.207	0.191	0.069	0.104	0.183	0.196
1940	0.209	0.179	0.117	0.093	0.078	0.157	0.133	0.195	0.224	0.221
1950	0.161	0.164	0.137	0.139	0.142	0.128	0.113	0.093	0.135	0.092
1960	0.109	0.110	0.093	0.127	0.102	0.082	0.061	0.074	0.043	0.020
1970	0.037	0.021	0.034	0.062	0.053	0.037	0.044	0.031	0.044	0.030
1980	0.024	0.008								

AREA	SIG14-2									
1880	11.046	9.286	8.591	8.496	8.198	11.381	32.602	17.882	12.644	19.689
1890	8.954	7.277	7.868	13.693	15.111	11.800	13.951	17.884	17.059	17.584
1900	1.484	2.792	5.371	6.748	10.095	9.403	14.518	15.027	20.358	15.895
1910	8.319	5.496	8.019	13.956	11.387	11.990	15.480	17.946	16.747	14.738
1920	22.997	29.455	19.309	16.537	15.155	14.257	16.514	16.722	27.546	26.267
1930	16.806	12.288	9.430	21.129	20.807	19.437	7.078	10.725	19.037	20.623
1940	22.257	19.280	12.711	10.165	8.567	17.360	14.828	21.941	25.498	25.466
1950	18.745	19.262	16.220	16.578	17.061	15.487	13.758	11.383	16.621	11.392
1960	13.566	13.766	11.698	16.063	12.974	10.478	7.822	9.520	5.548	2.584

Output file KNOB.IND, continued

1970	4.788	2.721	4.411	8.063	6.912	4.836	5.762	4.067	5.782	3.950
1980	3.164	1.055								

SPLINE	SIG14-2									
1880	12.040	12.069	12.097	12.126	12.152	12.176	12.195	12.208	12.215	12.218
1890	12.218	12.216	12.215	12.216	12.220	12.227	12.239	12.256	12.282	12.316
1900	12.362	12.424	12.502	12.598	12.712	12.843	12.990	13.152	13.326	13.513
1910	13.712	13.923	14.144	14.375	14.614	14.857	15.103	15.347	15.588	15.821
1920	16.045	16.257	16.455	16.638	16.807	16.961	17.100	17.224	17.332	17.423
1930	17.499	17.560	17.602	17.643	17.667	17.679	17.680	17.670	17.650	17.618
1940	17.574	17.516	17.445	17.361	17.263	17.149	17.018	16.867	16.694	16.497
1950	16.275	16.027	15.755	15.459	15.140	14.799	14.438	14.057	13.658	13.242
1960	12.810	12.363	11.903	11.431	10.947	10.454	9.955	9.450	8.943	8.434
1970	7.925	7.417	6.910	6.405	5.900	5.396	4.892	4.387	3.883	3.379
1980	2.874	2.369								

INDEX	SIG14-2									
1880	0.917	0.769	0.710	0.701	0.675	0.935	2.673	1.465	1.035	1.611
1890	0.733	0.596	0.644	1.121	1.237	0.965	1.140	1.459	1.389	1.428
1900	0.120	0.225	0.430	0.536	0.794	0.732	1.118	1.143	1.528	1.176
1910	0.607	0.395	0.567	0.971	0.779	0.807	1.025	1.169	1.074	0.932
1920	1.433	1.812	1.173	0.994	0.902	0.841	0.966	0.971	1.589	1.508
1930	0.960	0.700	0.536	1.198	1.178	1.099	0.400	0.607	1.079	1.171
1940	1.266	1.101	0.729	0.585	0.496	1.012	0.871	1.301	1.527	1.544
1950	1.152	1.202	1.030	1.072	1.127	1.046	0.953	0.810	1.217	0.860
1960	1.059	1.113	0.983	1.405	1.185	1.002	0.786	1.007	0.620	0.306
1970	0.604	0.367	0.638	1.259	1.171	0.896	1.178	0.927	1.489	1.169
1980	1.101	0.446								

RING WIDTHS	SIG16-1 MS= 0.357 1862= 0.00 CM.									
1862	0.265	0.424	0.241	0.212	0.159	0.205	0.210	0.198		
1870	0.025	0.068	0.161	0.197	0.045	0.066	0.160	0.133	0.250	0.191
1880	0.294	0.194	0.281	0.244	0.229	0.172	0.266	0.220	0.177	0.215
1890	0.049	0.106	0.188	0.219	0.255	0.152	0.172	0.196	0.245	0.202
1900	0.102	0.157	0.131	0.137	0.145	0.182	0.169	0.210	0.186	0.217
1910	0.122	0.094	0.133	0.162	0.081	0.116	0.120	0.134	0.165	0.147
1920	0.153	0.166	0.108	0.142	0.128	0.132	0.130	0.110	0.153	0.148
1930	0.094	0.039	0.092	0.152	0.113	0.122	0.086	0.123	0.173	0.173
1940	0.149	0.168	0.097	0.084	0.058	0.137	0.094	0.095	0.121	0.136
1950	0.094	0.085	0.071	0.078	0.054	0.082	0.100	0.063	0.128	0.080
1960	0.090	0.090	0.099	0.090	0.054	0.053	0.035	0.054	0.057	0.066
1970	0.089	0.067	0.057	0.070	0.053	0.066	0.044	0.014	0.077	0.037
1980	0.041	0.044								

AREA	SIG16-1									
1862	8.456	14.447	8.715	7.968	6.161	8.178	8.652	8.411		
1870	1.080	2.956	7.115	8.928	2.073	3.064	7.542	6.392	12.315	9.673
1880	15.338	10.418	15.510	13.870	13.358	10.249	16.217	13.748	11.282	13.969
1890	3.224	7.026	12.636	14.999	17.845	10.831	12.431	14.393	18.330	15.397
1900	7.872	12.244	10.335	10.924	11.690	14.860	13.985	17.628	15.845	18.760
1910	10.677	8.291	11.825	14.554	7.339	10.581	11.035	12.430	15.460	13.918
1920	14.630	16.039	10.528	13.954	12.687	13.191	13.098	11.166	15.658	15.286
1930	9.780	4.074	9.648	16.057	12.031	13.080	9.276	13.348	18.935	19.123
1940	16.621	18.907	10.998	9.571	6.635	15.756	10.879	11.051	14.157	16.022
1950	11.142	10.123	8.491	9.364	6.505	9.913	12.147	7.685	15.690	9.859
1960	11.139	11.190	12.368	11.297	6.802	6.694	4.430	6.851	7.251	8.422
1970	11.400	8.615	7.351	9.056	6.877	8.588	5.741	1.829	10.082	4.858
1980	5.393	5.800								

SPLINE	SIG16-1									
1862	7.044	7.174	7.304	7.436	7.569	7.707	7.849	7.997		
1870	8.152	8.316	8.489	8.671	8.861	9.059	9.264	9.474	9.687	9.901
1880	10.114	10.323	10.527	10.724	10.914	11.096	11.270	11.435	11.592	11.741
1890	11.883	12.019	12.147	12.268	12.380	12.483	12.578	12.663	12.740	12.809
1900	12.869	12.923	12.970	13.012	13.048	13.078	13.102	13.121	13.134	13.141
1910	13.144	13.144	13.143	13.140	13.138	13.136	13.134	13.132	13.131	13.128
1920	13.124	13.119	13.112	13.104	13.094	13.084	13.072	13.060	13.046	13.031
1930	13.015	12.998	12.981	12.962	12.940	12.913	12.881	12.843	12.796	12.740
1940	12.673	12.596	12.509	12.413	12.309	12.198	12.079	11.954	11.822	11.684
1950	11.540	11.391	11.238	11.081	10.922	10.760	10.595	10.427	10.256	10.080

Output file KNOB.IND, continued

1960	9.900	9.717	9.530	9.340	9.147	8.953	8.758	8.563	8.367	8.170
1970	7.973	7.773	7.573	7.371	7.167	6.962	6.757	6.551	6.345	6.139
1980	5.933	5.727								

INDEX		SIG16-1								
1862	1.200	2.014	1.193	1.072	0.814	1.061	1.102	1.052		
1870	0.132	0.355	0.838	1.030	0.234	0.338	0.814	0.675	1.271	0.977
1880	1.517	1.009	1.473	1.293	1.224	0.924	1.439	1.202	0.973	1.190
1890	0.271	0.585	1.040	1.223	1.441	0.868	0.988	1.137	1.439	1.202
1900	0.612	0.948	0.797	0.840	0.896	1.136	1.067	1.344	1.206	1.428
1910	0.812	0.631	0.900	1.108	0.559	0.806	0.840	0.946	1.177	1.060
1920	1.115	1.223	0.803	1.065	0.969	1.008	1.002	0.855	1.200	1.173
1930	0.751	0.313	0.743	1.239	0.930	1.013	0.720	1.039	1.480	1.501
1940	1.311	1.501	0.879	0.771	0.539	1.292	0.901	0.924	1.198	1.371
1950	0.966	0.889	0.756	0.845	0.596	0.921	1.146	0.737	1.530	0.978
1960	1.125	1.152	1.298	1.210	0.744	0.748	0.506	0.800	0.867	1.031
1970	1.430	1.108	0.971	1.229	0.960	1.234	0.850	0.279	1.589	0.791
1980	0.909	1.013								

RING WIDTHS		SIG16-2		MS=		0.410		1850=		0.00 CM.	
1850	0.207	0.312	0.251	0.203	0.198	0.297	0.188	0.301	0.233	0.200	
1860	0.026	0.088	0.248	0.182	0.149	0.151	0.207	0.267	0.222	0.169	
1870	0.020	0.034	0.112	0.159	0.059	0.078	0.179	0.112	0.274	0.164	
1880	0.216	0.132	0.179	0.135	0.135	0.135	0.257	0.117	0.165	0.195	
1890	0.042	0.048	0.122	0.186	0.201	0.112	0.105	0.138	0.178	0.107	
1900	0.031	0.125	0.091	0.100	0.089	0.101	0.160	0.187	0.234	0.150	
1910	0.081	0.089	0.101	0.110	0.071	0.092	0.092	0.138	0.169	0.120	
1920	0.132	0.128	0.120	0.126	0.120	0.092	0.107	0.110	0.117	0.143	
1930	0.089	0.048	0.061	0.070	0.087	0.097	0.046	0.086	0.147	0.129	
1940	0.118	0.088	0.071	0.071	0.071	0.129	0.106	0.115	0.135	0.093	
1950	0.103	0.093	0.079	0.115	0.086	0.110	0.089	0.072	0.126	0.077	
1960	0.085	0.090	0.061	0.077	0.049	0.039	0.027	0.032	0.032	0.034	
1970	0.053	0.041	0.029	0.037	0.034	0.068	0.035	0.012	0.071	0.039	
1980	0.041	0.030									

AREA		SIG16-2								
1850	3.329	5.526	4.890	4.244	4.389	7.045	4.746	8.061	6.631	5.964
1860	0.794	2.718	7.922	6.060	5.116	5.327	7.535	10.117	8.753	6.871
1870	0.825	1.408	4.690	6.794	2.561	3.420	7.993	5.103	12.817	7.897
1880	10.659	6.658	9.204	7.075	7.189	7.304	14.221	6.611	9.470	11.412
1890	2.489	2.858	7.330	11.356	12.516	7.084	6.713	8.928	11.693	7.125
1900	2.078	8.439	6.205	6.879	6.175	7.068	11.328	13.443	17.132	11.163
1910	6.087	6.735	7.704	8.463	5.503	7.178	7.231	10.946	13.568	9.743
1920	10.822	10.598	10.029	10.628	10.215	7.893	9.246	9.581	10.274	12.674
1930	7.953	4.310	5.498	6.338	7.920	8.886	4.235	7.953	13.702	12.136
1940	11.192	8.404	6.816	6.848	6.879	12.580	10.415	11.379	13.464	9.342
1950	10.410	9.457	8.076	11.826	8.898	11.449	9.319	7.575	13.335	8.198
1960	9.093	9.678	6.588	8.350	5.333	4.255	2.952	3.504	3.511	3.737
1970	5.840	4.530	3.210	4.104	3.778	7.579	3.912	1.343	7.965	4.389
1980	4.624	3.390								

SPLINE		SIG16-2								
1850	4.634	4.714	4.794	4.873	4.953	5.031	5.110	5.187	5.265	5.342
1860	5.419	5.498	5.577	5.657	5.738	5.819	5.901	5.984	6.067	6.152
1870	6.239	6.328	6.422	6.518	6.617	6.719	6.822	6.925	7.027	7.127
1880	7.224	7.317	7.407	7.493	7.574	7.651	7.724	7.792	7.857	7.918
1890	7.976	8.033	8.088	8.142	8.194	8.244	8.293	8.341	8.389	8.437
1900	8.486	8.536	8.587	8.639	8.692	8.744	8.794	8.843	8.888	8.930
1910	8.968	9.005	9.040	9.074	9.106	9.138	9.168	9.196	9.221	9.244
1920	9.262	9.277	9.288	9.297	9.303	9.306	9.308	9.309	9.310	9.310
1930	9.311	9.314	9.319	9.327	9.338	9.350	9.364	9.379	9.392	9.404
1940	9.412	9.416	9.415	9.409	9.397	9.377	9.349	9.310	9.260	9.199
1950	9.125	9.039	8.941	8.829	8.706	8.569	8.420	8.260	8.088	7.906
1960	7.715	7.516	7.310	7.100	6.886	6.671	6.456	6.242	6.030	5.820
1970	5.614	5.411	5.210	5.012	4.817	4.624	4.432	4.241	4.051	3.862
1980	3.672	3.483								

INDEX		SIG16-2								
1850	0.718	1.172	1.020	0.871	0.886	1.400	0.929	1.554	1.260	1.116
1860	0.146	0.494	1.420	1.071	0.892	0.915	1.277	1.691	1.443	1.117

Output file KNOB.IND, continued

1870	0.132	0.223	0.730	1.042	0.387	0.509	1.172	0.737	1.824	1.108
1880	1.476	0.910	1.243	0.944	0.949	0.955	1.841	0.848	1.205	1.441
1890	0.312	0.356	0.906	1.395	1.528	0.859	0.809	1.070	1.394	0.844
1900	0.245	0.989	0.723	0.796	0.710	0.808	1.288	1.520	1.927	1.250
1910	0.679	0.748	0.852	0.933	0.604	0.785	0.789	1.190	1.471	1.054
1920	1.168	1.142	1.080	1.143	1.098	0.848	0.993	1.029	1.104	1.361
1930	0.854	0.463	0.590	0.679	0.848	0.950	0.452	0.848	1.459	1.291
1940	1.189	0.893	0.724	0.728	0.732	1.342	1.114	1.222	1.454	1.016
1950	1.141	1.046	0.903	1.339	1.022	1.336	1.107	0.917	1.649	1.037
1960	1.179	1.288	0.901	1.176	0.774	0.638	0.457	0.561	0.582	0.642
1970	1.040	0.837	0.616	0.819	0.784	1.639	0.883	0.317	1.966	1.136
1980	1.259	0.973								

Output file KNOB.IX1

SIGNAL KNOB PITCH PINE

CRNIDX1900	494	23	745	23	703	23	948	23	941	23	709	23	888	231163	231385	231174	23	
CRNIDX1910	803	24	713	24	960	241122	24	741	24	843	24	973	241122	241107	241083	24		
CRNIDX1920	1219	241242	241048	24	981	24	914	24	903	24	936	24	922	241324	241356	24		
CRNIDX1930	761	24	422	24	517	24	898	24	906	24	999	24	694	24	959	241237	241432	24
CRNIDX1940	1215	241222	24	801	24	713	24	488	241114	24	981	241104	241453	241312	24			
CRNIDX1950	1306	241158	24	980	241067	24	993	241132	241111	24	824	241354	24	897	24			
CRNIDX1960	1077	241051	24	970	241244	24	823	24	773	24	684	241246	241055	24	769	24		
CRNIDX1970	939	24	536	24	625	241254	241175	241153	24	984	24	701	241271	24	734	24		
CRNIDX1980	1032	24	664	249990	09990	09990	09990	09990	09990	09990	09990	09990	09990	09990	09990	0		

Output file KNOB.IX2

```

SIGNAL KNOB PITCH PINE
SIG01-1900 911 0 1237 0 866 02213 01484 0 227 0 521 01135 01473 01533 0
SIG01-1910 746 0 562 01028 01563 0 920 0 461 0 776 0 753 0 886 01356 0
SIG01-19201132 01068 0 815 0 697 0 729 0 925 0 909 01066 01365 01373 0
SIG01-1930 573 0 460 0 468 0 896 0 695 0 837 0 730 01008 01315 01428 0
SIG01-1940 997 01308 0 776 0 784 0 448 01333 01215 01245 01424 01176 0
SIG01-19501122 01191 01000 0 859 0 956 01264 01162 0 914 01138 0 971 0
SIG01-19601302 01227 0 858 01112 0 894 0 687 0 805 01794 01262 0 886 0
SIG01-1970 922 0 488 0 379 01252 01002 0 764 0 834 0 636 01182 0 909 0
SIG01-19801148 0 583 09990 09990 09990 09990 09990 09990 09990 09990 0
SIG01-1890 866 01011 01357 01039 01145 0 844 01074 01313 02173 01304 0
SIG01-1900 829 01086 0 756 01329 01517 0 110 0 361 0 833 01376 01381 0
SIG01-1910 639 0 386 0 749 01489 0 978 0 407 0 514 0 709 0 650 01081 0
SIG01-19201000 0 933 0 900 01097 0 996 01173 01051 0 962 01311 01157 0
SIG01-1930 593 0 278 0 351 0 734 0 804 01221 01000 01293 01098 01543 0
SIG01-19401287 01193 0 604 0 848 0 501 01413 01226 01217 01751 01138 0
SIG01-1950 921 0 909 0 909 01003 01129 01127 0 886 0 676 01192 01105 0
SIG01-19601570 01341 0 701 0 976 0 761 0 733 0 630 01668 01338 0 996 0
SIG01-1970 855 0 421 0 565 01415 01221 01226 0 751 0 501 01150 0 681 0
SIG01-19801065 0 606 09990 09990 09990 09990 09990 09990 09990 09990 0
SIG02-18501000 01003 01049 01250 01090 0 956 0 686 0 946 01143 01042 0
SIG02-1860 458 0 842 01342 0 814 01065 01813 0 805 01844 01587 01220 0
SIG02-1870 206 0 828 0 731 0 709 0 164 0 320 0 436 0 664 01517 0 941 0
SIG02-18801395 0 637 01279 01010 01318 0 735 01264 01052 01827 02333 0
SIG02-1890 657 0 822 01057 01241 01576 0 374 0 575 01306 0 725 0 831 0
SIG02-1900 551 01171 01067 0 929 0 941 0 201 0 891 01208 01555 01147 0
SIG02-1910 398 0 445 0 863 0 830 0 732 0 577 0 968 01201 01301 01096 0
SIG02-19201188 01317 01054 01632 01542 0 942 0 814 0 881 01429 01930 0
SIG02-1930 733 0 0 0 757 01028 0 837 0 927 0 836 0 722 01381 01620 0
SIG02-19401257 0 940 0 851 0 826 0 556 01177 0 892 01019 01519 01275 0
SIG02-19501487 01224 0 731 0 992 0 547 0 820 0 890 0 582 01179 0 896 0
SIG02-19601045 0 592 0 785 01388 0 592 0 735 0 723 02222 0 991 0 141 0
SIG02-1970 993 0 165 0 225 02431 01267 01607 0 281 01142 01456 02997 0
SIG02-1980 0 0 0 09990 09990 09990 09990 09990 09990 09990 09990 0
SIG02-18439990 09990 09990 0 498 0 795 0 712 0 868 01207 01346 0 863 0
SIG02-1850 909 01012 01424 01597 01269 01017 0 913 01251 01397 0 900 0
SIG02-1860 337 0 822 01020 0 739 01090 01937 0 739 01533 01090 01232 0
SIG02-1870 264 0 809 0 491 0 660 0 174 0 208 0 368 0 868 01590 01515 0
SIG02-18801515 0 844 01109 01227 01200 01038 01318 01105 01491 02442 0
SIG02-1890 763 01288 0 974 01268 01439 0 462 0 628 0 887 0 565 0 523 0
SIG02-1900 454 0 989 01261 0 626 0 854 0 180 0 784 01187 01355 01150 0
SIG02-1910 519 0 578 0 912 0 608 0 811 0 694 01201 01683 01704 01431 0
SIG02-19201230 01659 01149 0 840 0 885 0 723 0 678 0 766 01002 01520 0
SIG02-1930 931 0 0 0 661 0 703 0 746 01272 0 745 0 818 01404 01512 0
SIG02-19401342 01185 0 938 0 882 0 586 01075 0 901 01111 01596 0 984 0
SIG02-19501564 01199 0 819 0 938 0 783 0 909 01115 01071 01472 01100 0
SIG02-19601553 01139 01095 01327 0 486 0 355 0 372 0 755 01150 0 530 0
SIG02-1970 793 0 492 0 677 02225 01094 01620 0 313 01041 01102 01770 0
SIG02-1980 0 0 0 09990 09990 09990 09990 09990 09990 09990 09990 0
SIG03-18901850 0 870 0 736 01094 01088 01642 0 379 0 570 01317 01742 0
SIG03-1900 996 0 263 0 977 0 929 01265 01253 0 503 0 927 01126 01099 0
SIG03-1910 214 0 272 0 804 01171 0 633 01071 01193 01237 01127 01024 0
SIG03-19201198 01005 0 937 01106 0 979 0 954 0 969 0 857 01603 01488 0
SIG03-1930 574 0 181 0 308 0 981 01201 01134 0 694 01000 01508 01958 0
SIG03-19401474 01561 0 644 0 728 0 266 0 931 01101 01124 01600 01314 0
SIG03-1950 618 01203 0 634 01359 01064 0 880 01356 0 960 01150 0 580 0
SIG03-1960 750 0 775 01194 01259 0 719 0 949 0 974 01163 0 647 0 535 0
SIG03-1970 895 0 405 0 408 01650 0 834 01900 01437 0 902 01163 0 242 0
SIG03-19801522 0 663 09990 09990 09990 09990 09990 09990 09990 09990 0
SIG03-1880 593 01062 01269 01081 0 515 01031 01715 01000 01000 01565 0
SIG03-1890 755 0 738 01172 01233 01912 0 630 0 478 01537 01133 01055 0
SIG03-1900 325 0 808 0 636 01267 01444 0 147 0 732 01099 01335 01320 0
SIG03-1910 310 0 247 0 563 0 762 0 666 0 951 01390 01295 01300 01423 0
SIG03-19201293 01291 0 989 0 967 0 966 0 848 0 846 0 700 01642 01522 0
SIG03-1930 701 0 146 0 379 0 833 0 946 0 860 0 423 0 679 01237 02086 0
SIG03-19401616 02025 0 918 01038 0 405 01034 01175 0 805 01360 01258 0
SIG03-19501159 01035 0 661 01150 01465 0 900 01429 01064 01412 0 401 0
SIG03-1960 672 0 722 0 869 01249 0 400 0 475 0 495 0 855 0 944 0 775 0
SIG03-19701144 0 774 0 669 01580 01059 01333 01565 0 832 0 943 0 274 0

```

Output file KNOB.IX2, continued

SIG03-19802133	0 531	09990	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG05-1910 327	0 520	0 569	01044	0 723	0 661	0 828	01027	0 987	0 943	0	0
SIG05-19201473	01549	01531	01195	0 978	01012	0 882	01208	01417	01264	0	0
SIG05-1930 523	0 466	0 397	0 756	0 749	01000	0 771	0 916	01034	01157	0	0
SIG05-19401040	0 959	0 537	0 331	0 327	0 730	0 847	01024	01254	02231	0	0
SIG05-19502529	01334	01057	0 967	0 812	01102	0 976	0 863	01313	0 779	0	0
SIG05-1960 841	0 951	01175	01773	01076	0 886	0 706	01506	01323	0 951	0	0
SIG05-1970 811	0 698	0 523	0 917	0 931	0 657	0 891	0 538	01248	0 560	0	0
SIG05-1980 535	0 341	09990	09990	09990	09990	09990	09990	09990	09990	0	0
SIG05-1880 845	01301	0 887	0 646	01110	01316	01720	01333	01780	02535	0	0
SIG05-1890 361	0 141	0 816	01257	01368	0 768	0 901	01496	01656	01714	0	0
SIG05-1900 232	0 541	0 529	0 958	01171	0 109	0 358	0 929	01373	0 852	01098	0
SIG05-1910 733	0 599	0 691	0 683	0 630	0 668	0 929	01373	0 852	01098	0	0
SIG05-19201058	01009	0 783	0 998	0 781	01134	0 997	01038	01431	01230	0	0
SIG05-1930 852	0 971	0 855	01341	01049	0 984	0 652	0 912	01144	01043	0	0
SIG05-19401158	01298	01281	0 764	0 683	0 846	0 932	0 934	01511	0 894	0	0
SIG05-19501007	01150	0 986	0 978	01030	01388	01316	0 764	01220	0 756	0	0
SIG05-19601258	01164	01110	01091	0 912	01091	0 870	01371	01083	0 719	0	0
SIG05-1970 727	0 331	0 541	01131	01450	01114	0 802	0 579	01819	0 439	0	0
SIG05-1980 444	0 180	09990	09990	09990	09990	09990	09990	09990	09990	0	0
SIG06-18829990	09990	01585	01013	0 845	0 718	01376	01102	01110	01826	0	0
SIG06-1890 269	0 300	0 540	0 742	01000	0 611	01007	01384	01171	01050	0	0
SIG06-1900 621	0 994	0 867	01252	0 888	0 609	01144	01330	01700	0 941	0	0
SIG06-1910 803	0 836	0 923	01282	0 710	01020	01129	01365	01130	01014	0	0
SIG06-1920 914	0 823	0 876	01267	01103	01131	01097	0 815	01309	01337	0	0
SIG06-1930 702	0 0	0 530	0 935	01105	01212	0 624	0 965	01231	01615	0	0
SIG06-19401400	01124	0 717	0 598	0 466	0 902	01033	01271	01363	01204	0	0
SIG06-19501339	01113	01375	01066	01051	01047	01042	0 710	01284	0 593	0	0
SIG06-19601045	0 816	0 883	01280	0 742	0 693	0 733	01067	0 520	0 661	0	0
SIG06-19701122	0 773	0 648	02053	01421	01424	0 760	0 950	01401	0 560	0	0
SIG06-1980 802	0 696	09990	09990	09990	09990	09990	09990	09990	09990	0	0
SIG06-1880 706	0 856	01120	0 709	0 723	0 860	02856	01422	01088	01294	0	0
SIG06-1890 95	0 304	0 565	0 613	01173	0 773	0 894	01335	01196	0 979	0	0
SIG06-1900 574	0 986	0 799	01646	01173	0 799	01083	01305	01533	01105	0	0
SIG06-1910 758	0 755	01001	01138	0 607	0 962	01063	01283	01021	01135	0	0
SIG06-1920 902	0 958	0 862	01008	01134	01240	0 933	0 728	01351	01317	0	0
SIG06-1930 708	0 0	0 524	0 755	01069	01168	0 799	01077	01353	01734	0	0
SIG06-19401288	01094	0 431	0 733	0 447	01019	01322	01302	01251	01021	0	0
SIG06-19501399	01101	01066	0 951	01050	01001	01130	0 887	01319	0 815	0	0
SIG06-19601306	01166	0 977	0 917	0 637	0 668	0 685	0 953	0 947	0 791	0	0
SIG06-1970 796	0 782	0 786	01471	01042	01190	01045	0 621	01666	0 794	0	0
SIG06-19801185	0 753	09990	09990	09990	09990	09990	09990	09990	09990	0	0
SIG08-18879990	09990	09990	09990	09990	09990	09990	01269	01078	02249	0	0
SIG08-1890 465	0 559	0 777	0 918	0 981	0 582	0 848	01217	01188	0 859	0	0
SIG08-1900 707	0 899	0 903	01173	01008	0 826	0 839	01328	01520	01017	0	0
SIG08-19101027	0 968	0 976	01146	0 876	0 827	01160	0 926	01145	01020	0	0
SIG08-19201261	01252	01040	0 785	0 741	0 508	0 788	0 683	01077	01319	0	0
SIG08-1930 803	01027	0 744	01032	0 983	0 737	0 745	0 965	0 881	01284	0	0
SIG08-19401211	01452	0 683	0 669	0 515	01463	0 827	0 836	01416	01251	0	0
SIG08-19501530	01141	01042	01248	0 761	01126	01052	0 616	01435	01044	0	0
SIG08-19601002	01225	0 839	01249	0 941	0 943	0 809	01401	01215	0 586	0	0
SIG08-19701038	0 378	0 907	01372	01487	0 547	0 911	0 876	01318	0 419	0	0
SIG08-1980 998	0 633	09990	09990	09990	09990	09990	09990	09990	09990	0	0
SIG08-1890 321	0 523	0 959	01080	01120	0 667	01010	01454	01465	01301	0	0
SIG08-1900 867	01169	0 994	01004	01311	0 882	01385	01081	01143	01139	0	0
SIG08-1910 863	0 663	0 877	0 771	0 620	0 851	01061	0 915	01275	01234	0	0
SIG08-19201221	01277	0 966	0 751	0 584	0 610	0 852	0 794	01196	01372	0	0
SIG08-1930 969	0 727	0 735	01048	0 932	0 894	0 638	01105	01108	01521	0	0
SIG08-1940 931	0 988	0 690	0 665	0 420	01198	01303	01212	01642	01256	0	0
SIG08-19501241	0 914	01190	01226	0 878	01286	01264	0 712	01060	0 666	0	0
SIG08-1960 927	01105	0 751	01132	01060	0 910	0 873	01360	01198	0 552	0	0
SIG08-19701070	0 460	0 922	01414	01995	0 711	01127	0 845	0 923	0 330	0	0
SIG08-1980 981	0 335	09990	09990	09990	09990	09990	09990	09990	09990	0	0
SIG09-1890 982	01004	01050	01433	01493	0 570	0 515	0 677	01125	01027	0	0
SIG09-1900 752	01081	0 727	01202	0 841	0 371	0 895	01200	01382	0 836	0	0
SIG09-19101116	0 981	01187	01465	0 608	01088	01001	01533	01206	01052	0	0
SIG09-19201512	01178	0 761	0 668	01032	01036	0 910	0 940	01391	01298	0	0
SIG09-1930 628	0 0	0 119	0 613	0 616	0 868	0 850	01278	01534	01477	0	0

Output file KNOB.IX2, continued

SIG09-19401457	01495	0 834	0 992	0 350	01285	01221	01084	01283	0 962	0
SIG09-19501017	01171	0 810	01125	0 940	01307	0 916	0 766	01194	0 871	0
SIG09-19601114	0 966	01091	01198	0 636	0 689	0 643	01735	01625	01086	0
SIG09-19701216	0 0	0 969	0 814	0 569	01168	0 835	0 562	01219	0 531	0
SIG09-19801242	01216	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG09-18901445	01276	01435	01239	01249	01189	0 410	0 670	0 627	0 982	0
SIG09-1900 761	0 624	01010	0 745	01013	0 451	0 575	0 773	01189	01009	0
SIG09-19101202	01188	01129	01223	01219	0 583	0 860	0 838	01093	0 898	0
SIG09-19201211	01233	01428	0 995	01033	0 964	0 926	01147	0 743	01209	0
SIG09-19301191	0 715	0 348	0 566	0 630	0 900	0 841	01039	01097	01262	0
SIG09-19401117	01282	01222	0 686	0 538	01402	0 982	01511	01702	01271	0
SIG09-19501280	01178	0 913	0 965	0 981	01108	0 907	0 701	01310	0 712	0
SIG09-19601053	0 932	0 995	0 941	0 600	0 546	0 638	01068	01450	01255	0
SIG09-19701140	0 868	0 163	01091	0 908	01355	01043	0 587	0 882	0 971	0
SIG09-19801347	01105	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG11-18639990	09990	09990	0 659	0 873	0 601	01002	0 992	01177	01234	0
SIG11-18701101	0 76	0 284	0 409	0 379	0 577	0 919	0 913	01362	0 990	0
SIG11-18801367	01058	01879	01168	01373	0 914	01345	0 989	01172	01316	0
SIG11-18901110	01074	01031	01139	01237	0 790	0 769	01049	01112	0 785	0
SIG11-1900 584	0 685	0 563	01196	0 897	0 908	0 991	01123	01215	01516	0
SIG11-19101002	0 862	01101	01220	0 772	01004	01064	01365	0 992	01006	0
SIG11-19201133	01354	01227	0 650	0 689	0 762	0 930	0 911	01562	01384	0
SIG11-1930 757	0 0	0 270	0 695	0 414	0 721	0 366	0 692	01090	01295	0
SIG11-19401182	01228	0 826	0 895	0 621	01336	0 888	01231	01466	01535	0
SIG11-19501332	01370	01130	01006	01159	01178	01226	0 937	01635	01011	0
SIG11-19601085	0 952	01041	01158	0 696	0 763	0 364	01216	0 756	0 653	0
SIG11-1970 759	0 366	0 704	01072	01540	01872	01940	0 526	01072	0 885	0
SIG11-1980 561	0 0	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG11-1870 39	0 213	0 296	0 716	0 560	0 761	0 916	01112	01869	01290	0
SIG11-18801229	01104	01929	01134	01136	0 711	01425	01327	01481	01411	0
SIG11-1890 925	0 727	0 882	01169	01340	0 796	0 494	01150	01004	0 768	0
SIG11-1900 284	0 434	0 567	01067	01014	0 725	0 721	01489	02002	01642	0
SIG11-1910 899	01029	01194	01422	0 589	01006	01088	01125	01000	01138	0
SIG11-19201134	0 900	0 753	0 938	0 806	0 645	0 965	0 957	01330	01300	0
SIG11-1930 657	0 0	0 306	0 601	0 666	0 910	0 473	0 651	0 827	01396	0
SIG11-19401090	01429	01124	0 843	0 526	01325	0 824	0 974	01502	01636	0
SIG11-19501457	01443	01030	01365	0 960	01065	01391	0 946	01452	01038	0
SIG11-1960 710	0 990	0 785	01281	0 774	01002	0 563	01103	0 713	0 517	0
SIG11-19701153	0 263	0 501	01228	01346	01578	01389	0 896	0 902	01099	0
SIG11-19801113	0 409	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG12-18829990	09990	0 867	01145	01284	01478	01054	01619	01854	01957	0
SIG12-1890 372	0 400	0 694	0 797	01042	0 210	0 196	0 448	0 963	0 783	0
SIG12-1900 139	0 632	0 557	0 928	0 933	01406	01369	01019	01179	01085	0
SIG12-19101360	01110	01935	01282	0 957	01446	0 836	01283	01417	01203	0
SIG12-19201705	01400	01201	0 316	0 345	0 537	0 890	0 759	01244	01290	0
SIG12-1930 699	0 158	0 387	0 549	01015	01113	01134	01300	01456	01535	0
SIG12-19401139	0 981	01049	0 749	0 453	0 651	0 343	0 420	01433	01325	0
SIG12-19501117	01402	01075	01056	01111	01038	01094	0 748	01685	01393	0
SIG12-19601207	01252	0 728	01759	0 789	0 486	0 583	0 883	01116	0 999	0
SIG12-1970 867	0 327	0 426	0 813	01184	01417	0 887	0 835	01293	0 558	0
SIG12-19801430	01336	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG12-1900 147	0 464	0 249	0 530	0 482	0 876	01135	01203	01681	01275	0
SIG12-19101453	01116	01591	01698	0 969	01383	0 960	01411	01263	01196	0
SIG12-19201384	01242	01229	0 192	0 292	0 502	01014	0 828	01247	01233	0
SIG12-1930 474	0 179	0 423	0 807	01182	01034	0 882	01197	01114	01276	0
SIG12-1940 959	0 912	01177	0 721	0 391	0 959	0 938	01149	01395	01277	0
SIG12-19501128	0 955	01259	01288	01252	01284	0 874	0 632	01466	0 870	0
SIG12-19601214	01064	0 930	01453	0 715	0 670	0 661	01037	01238	0 908	0
SIG12-19701032	0 631	0 872	01296	01261	01256	0 590	0 609	01444	0 374	0
SIG12-1980 891	01106	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG13-1870 76	0 211	0 401	0 547	0 421	0 520	0 711	0 976	01338	01068	0
SIG13-18801219	01037	01710	01136	01371	01268	01852	01514	01626	01449	0
SIG13-1890 912	0 999	01139	01377	01396	0 927	0 782	01179	01091	0 917	0
SIG13-1900 539	0 816	0 438	0 606	0 569	0 605	0 570	0 924	01009	0 842	0
SIG13-1910 924	0 795	01011	01102	0 595	0 796	0 992	01057	0 884	0 782	0
SIG13-19201097	01290	01326	01180	01023	01055	01026	0 881	01163	01127	0
SIG13-1930 677	0 447	0 581	0 942	0 874	0 957	0 701	01056	01600	01406	0
SIG13-19401216	01279	0 804	0 810	0 570	01209	0 932	01193	01340	01084	0

Output file KNOB.IX2, continued

SIG13-19501438	01151	0 987	01097	01178	01076	01040	0 939	02023	01313	0
SIG13-19601009	0 815	0 736	0 889	0 731	0 554	0 506	01011	0 524	0 608	0
SIG13-1970 730	0 725	0 635	01202	01081	0 601	0 895	01084	01446	01676	0
SIG13-19801392	0 0	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG13-1870 109	0 191	0 320	0 592	0 590	0 561	0 687	0 833	01385	01247	0
SIG13-18801224	01177	01950	0 989	01200	0 897	01662	01458	01845	01852	0
SIG13-18901159	0 915	01032	01329	01692	0 922	01086	0 925	0 880	0 628	0
SIG13-1900 263	0 315	0 410	0 756	0 653	0 643	0 683	01100	01340	01063	0
SIG13-1910 798	0 595	0 981	01073	0 623	0 762	0 951	0 757	0 613	0 929	0
SIG13-19201196	01371	01116	01311	01120	0 933	01067	0 917	01415	01527	0
SIG13-1930 944	0 721	0 560	0 950	01013	01245	0 643	01022	01443	01387	0
SIG13-19401260	0 858	0 614	0 522	0 465	0 720	0 555	0 862	01065	01037	0
SIG13-19501098	01288	01087	01012	0 816	01367	01823	01419	01718	01129	0
SIG13-1960 950	01204	01579	01268	0 931	0 637	0 708	0 955	0 946	0 702	0
SIG13-1970 800	0 645	0 756	0 982	01254	0 580	0 217	01140	01097	01106	0
SIG13-1980 0	0 0	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG14-18801006	0 896	0 851	0 890	0 574	0 959	01426	01344	01303	01528	0
SIG14-1890 878	0 474	0 733	0 853	01018	0 972	01046	01768	01477	01437	0
SIG14-1900 244	0 442	0 449	0 828	01005	0 906	01097	01328	01274	01166	0
SIG14-1910 785	0 707	0 811	01077	0 738	0 848	0 854	0 950	01240	01491	0
SIG14-19201229	01153	0 912	0 816	0 702	0 938	0 867	01003	01552	01388	0
SIG14-1930 820	0 925	0 683	01039	01242	01098	0 537	0 842	01150	01532	0
SIG14-19401203	01118	0 555	0 490	0 523	0 872	0 996	01104	01534	01271	0
SIG14-19501381	01167	01051	01060	01362	01292	01039	0 760	01059	0 693	0
SIG14-1960 840	0 791	0 984	01204	0 887	0 930	0 771	0 893	0 968	0 574	0
SIG14-1970 850	0 542	0 919	01128	01178	01344	01294	0 974	01346	01083	0
SIG14-1980 978	0 455	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG14-1880 917	0 769	0 710	0 700	0 674	0 934	02673	01464	01035	01611	0
SIG14-1890 732	0 595	0 644	01120	01236	0 965	01139	01459	01388	01427	0
SIG14-1900 120	0 224	0 429	0 535	0 794	0 732	01117	01142	01527	01176	0
SIG14-1910 606	0 394	0 566	0 970	0 779	0 807	01024	01169	01074	0 931	0
SIG14-19201433	01811	01173	0 993	0 901	0 840	0 965	0 970	01589	01507	0
SIG14-1930 960	0 699	0 535	01197	01177	01099	0 400	0 606	01078	01170	0
SIG14-19401266	01100	0 728	0 585	0 496	01012	0 871	01300	01527	01543	0
SIG14-19501151	01201	01029	01072	01126	01046	0 952	0 809	01216	0 860	0
SIG14-19601059	01113	0 982	01405	01185	01002	0 785	01007	0 620	0 306	0
SIG14-1970 604	0 366	0 638	01258	01171	0 896	01177	0 926	01489	01169	0
SIG14-19801100	0 445	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG16-18629990	09990	01200	02013	01193	01071	0 814	01061	01102	01051	0
SIG16-1870 132	0 355	0 838	01029	0 234	0 338	0 814	0 674	01271	0 976	0
SIG16-18801516	01009	01473	01293	01223	0 923	01438	01202	0 973	01189	0
SIG16-1890 271	0 584	01040	01222	01441	0 867	0 988	01136	01438	01202	0
SIG16-1900 611	0 947	0 796	0 839	0 895	01136	01067	01343	01206	01427	0
SIG16-1910 812	0 630	0 899	01107	0 558	0 805	0 840	0 946	01177	01060	0
SIG16-19201114	01222	0 802	01064	0 968	01008	01001	0 855	01200	01173	0
SIG16-1930 751	0 313	0 743	01238	0 929	01012	0 720	01039	01479	01501	0
SIG16-19401311	01501	0 879	0 771	0 539	01291	0 900	0 924	01197	01371	0
SIG16-1950 965	0 888	0 755	0 845	0 595	0 921	01146	0 736	01529	0 978	0
SIG16-19601125	01151	01297	01209	0 743	0 747	0 505	0 800	0 866	01030	0
SIG16-19701429	01108	0 970	01228	0 959	01233	0 849	0 279	01588	0 791	0
SIG16-1980 908	01012	09990	09990	09990	09990	09990	09990	09990	09990	0
SIG16-1850 718	01172	01020	0 870	0 886	01400	0 928	01554	01259	01116	0
SIG16-1860 146	0 494	01420	01071	0 891	0 915	01276	01690	01442	01116	0
SIG16-1870 132	0 222	0 730	01042	0 387	0 508	01171	0 736	01823	01108	0
SIG16-18801475	0 909	01242	0 944	0 949	0 954	01841	0 848	01205	01441	0
SIG16-1890 312	0 355	0 906	01394	01527	0 859	0 809	01070	01393	0 844	0
SIG16-1900 244	0 988	0 722	0 796	0 710	0 808	01288	01520	01927	01250	0
SIG16-1910 678	0 747	0 852	0 932	0 604	0 785	0 788	01190	01471	01054	0
SIG16-19201168	01142	01079	01143	01098	0 848	0 993	01029	01103	01361	0
SIG16-1930 854	0 462	0 589	0 679	0 848	0 950	0 452	0 847	01458	01290	0
SIG16-19401189	0 892	0 723	0 727	0 732	01341	01114	01222	01453	01015	0
SIG16-19501140	01046	0 903	01339	01022	01336	01106	0 917	01648	01036	0
SIG16-19601178	01287	0 901	01176	0 774	0 637	0 457	0 561	0 582	0 642	0
SIG16-19701040	0 837	0 616	0 818	0 784	01639	0 882	0 316	01966	01136	0
SIG16-19801259	0 973	09990	09990	09990	09990	09990	09990	09990	09990	0

APPENDIX B
SOURCE-CODE LISTINGS FOR PROGRAMS

	Page
Program AREA.....	49
Program PLOT.....	65
Program MAKERAD.....	115
Program TRING.....	117

Source-code listing for program AREA

```

/*****
/* Program name   : AREA.C
/* Purpose       : Creates area and index files from a standard
/*                ring width file and radius data. Radius data
/*                may be created by AREA or from a MAKERAD file.
/* Input         : Standard ring width file in MM./100
/* Output        : Raw and smoothed BAI data for input to PLOT
/*                and standard index files.
/* Programmer    : Michael L. Field
/* Created for   : Tree-Ring Laboratory
/*                Mail Stop #461
/*                U.S. Geological Survey
/*                Reston, Virginia 22029
/* Files used    : BAREA.C
/*                GETRADS.C
/*                GETRINGS.C
/*                NORMAL.C
/*                SPLINE.C
/*                WRT_FLS.C
/*                COMMON.C
/* Date Created :
/* Modifications :
/* Purpose      :
/* Date        :
/* Programmer   :
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0
#define CR 013
#define LF 010

main()
{
    static double rba[1000];
    static double sba[1000];
    static double ringwidth[2000];
    static double lindex[1000];
    static int cores[1000];
    static double average[1000];
    static double means[1000];
    static double upper[1000];
    static double lower[1000];
    static double sum[1000][3];
    static double summ[1000][2];
    static double stdev[1000];

    static float ttable_vals[] = {
        0.0, 12.706, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262, 2.228,
        2.201, 2.179, 2.160, 2.145, 2.131, 2.120, 2.110, 2.101, 2.093, 2.086,
        2.080, 2.074, 2.069, 2.064, 2.060, 2.056, 2.052, 2.048, 2.045, 2.042};
    char rfile[41], radfile[41], bfile[41], crnfile[41], outfile[41], idx1file[41];
    char inline[81], id[9], idrw[9], idrad[9], title[81], temp_str[9];
    char ixfile[41], default_nam[41];
    char *cf, c, answ, *cp;
    int syr, eyr, sub1, file_len, rdone, elw;
    int i, j, k, basp, rad, form, qdone, ci, nyr, n2, tt;
    int first_yr, last_yr, year_cnt, crit_yr=0, canopy, canopy_year;
    int ind_file=FALSE, idx_file=FALSE, sta_file=FALSE;
    int all_ok = TRUE;
    double radius, spline_len, ttable, crit_val=0.0, ms, ymf, sdv;
    FILE *rwd, *rdd, *bd, *sd, *ixd, *ixd1;
    FILE *setupinf(), *setupout();
    char *fgets(), *gets(), *strcat(), *strcpy();
    char version[19];
    double atof(), fabs(), sqrt();

```

Source-code of program AREA, continued.

```

extern int index();

cls();
printf("\n\n\n\n\n");
printf("                                     BASAL AREA INCREMENT PROGRAM\n");
printf("                                     TREE-RING LABORATORY\n");
printf("                                     U.S. GEOLOGICAL SURVEY\n");
printf("                                     WATER RESOURCES DIVISION\n");
printf("                                     RESTON, VIRGINIA\n");
printf("                                     (Press enter to continue)");
gets(temp_str);
cls();
printf("          **** Press enter to select default answers ****\n");
printf("Enter the name of the ring-width(.RW) file: ");
gets(rwfile);
if ((rwd = setupinf(rwfile, "rb")) == (FILE *)NULL)
    exit();
elw = 0;
cp = rwfile;
i = 0;
while (i < 41 && *cp != '.' && *cp != '\0')
    default_nam[i++] = *cp++;
default_nam[i] = '\0';
fgets(title, 80, rwd);
i = strlen(title);
while ((c=title[i]) == ' ' || c == '\010' || c == '\013' || c == '\n')
    i--;
title[i-2] = '\0';
printf("\nThis is the included title:\n%s\n",title);
printf(" Title OK? (Y/N, default=Y)");
gets(temp_str);
if (temp_str[0] == 'n' || temp_str[0] == 'N')
{
    printf("Enter new title: ");
    gets(title);
}
printf("\nDoes the ring-width file contain Early and Late rings?");
printf("\n                                     (Y/N, default=N) ");
gets(temp_str);
if (temp_str[0] == 'y' || temp_str[0] == 'Y')
{
    printf("Do you want to process (E)arly or (L)ate rings (E/L)? ");
    gets(temp_str);
    if (temp_str[0] == 'e' || temp_str[0] == 'E')
        elw = 1;
    else
        elw = 2;
}
printf("\nOutput file name will be : %s\n",default_nam);
printf(" Name OK? (Y/N, default=Y)");
gets(temp_str);
if (temp_str[0] == 'n' || temp_str[0] == 'N')
{
    printf(" Enter the name of the output files: ");
    gets(default_nam);
}
file_len = index(default_nam, ".");
if (file_len != -1)
    default_nam[file_len] = '\0';
strcpy(crnfile, default_nam);
strcat(crnfile, ".crn");
strcpy(idx1file, default_nam);
strcat(idx1file, ".ix1");
printf("\nThe mean chronology file will be named %s.\n",crnfile);
if ((sd = setupout(crnfile, "w")) == (FILE *)NULL)
    exit();
fprintf(sd, "%s\n",title);
idx1 = fopen(idx1file, "w");
sta_file = TRUE;

```

Source-code of program AREA, continued.

```
printf("\nCreate individual cores(.IND) file? (Y/N, default=Y)");
gets(temp_str);
if (temp_str[0] != 'n' && temp_str[0] != 'N')
{
    strcpy(bafile, default_nam);
    strcat(bafile, ".ind");
    if ((bd = setupout(bafile, "w")) == (FILE *)NULL)
        exit();
    fprintf(bd, "%s\n", title);
    ind_file = TRUE;
}
printf("\nCreate index(.IX2) file? (Y/N, default=Y)");
gets(temp_str);
if (temp_str[0] != 'n' && temp_str[0] != 'N')
{
    strcpy(ixfile, default_nam);
    strcat(ixfile, ".ix2");
    if ((ixd = setupout(ixfile, "w")) == (FILE *)NULL)
        exit();
    fprintf(ixd, "%s\n", title);
    idx_file = TRUE;
}
canopy = FALSE;
canopy_year = 0;
printf("\nDo you want to use the estimated canopy radius(Y/N, default=N)? ");
gets(temp_str);
if (temp_str[0] == 'Y' || temp_str[0] == 'y')
{
    printf("\nDo you want trees (I)n the canopy or (U)nder the canopy? ");
    gets(temp_str);
    if (temp_str[0] == 'I' || temp_str[0] == 'i')
        canopy = 1;
    else
        canopy = 2;
    printf("\nRadius length to enter canopy (default=10 cm.): ");
    gets(temp_str);
    if ((i=strlen(temp_str)) > 0)
        crit_val = atof(temp_str);
    else
        crit_val = 10.0;
    printf("\nDo you want to use a year for the canopy value(Y/N, default=N)? ");
    gets(temp_str);
    if (temp_str[0] == 'Y' || temp_str[0] == 'y')
    {
        printf(" Enter canopy year: ");
        gets(temp_str);
        canopy_year = atoi(temp_str);
    }
}
spline_len = 60;
printf("\nEnter spline length in years(default=60 years): ");
gets(temp_str);
if (temp_str[0] != '\0')
    spline_len = atof(temp_str);
printf("\nEnter first year for mean chronology(.CRN) file: ");
gets(temp_str);
syr = atoi(temp_str);
printf("Enter last year for mean chronology(.CRN) file: ");
gets(temp_str);
eyr = atoi(temp_str);
n2 = eyr-syr+1;
rdone = FALSE;
qdone = FALSE;
while (!rdone)
{
    printf("\nYou must select a radius file from the following:\n\n");
    printf(" 1. Terminate\n 2. Enter old file name\n");
    printf(" 3. Create and use new file\n 4. Manually enter info\n\n");
    printf(" Choice: ");
```

Source-code of program AREA, continued.

```

while ((answ = getchar()) < '0' && answ > '5');
c = getchar();
if (answ == '1')
    exit();
if (answ != '4')
{
    if (answ == '2' || answ == '3')
    {
        printf("Enter radius file name(default=%s.RAD): ",default_nam);
        gets(radfile);
        if (strlen(radfile) < 2)
        {
            strcpy(radfile, default_nam);
            strcat(radfile, ".rad");
        }
    }
    if (answ == '2')
    {
        if ((rdd=setupinf(radfile, "r")) != (FILE *)NULL)
        {
            rdone = TRUE;
            cf = fgets(inline, 80, rdd); /* discard title of radius file */
        }
    }
    if (answ == '3')
    {
        if ((rdd=setupout(radfile, "w")) != (FILE *)NULL)
        {
            rdone = TRUE;
            fprintf(rdd, "%s\n", title);
            getrads(rwd, rdd, crit_val, elw);
            fclose(rdd);
            rdd = fopen(radfile, "r");
            fclose(rwd);
            rwd = fopen(rwfile, "r");
            cf = fgets(inline, 80, rdd); /* discard title of radius file */
            cf = fgets(inline, 80, rwd);
        }
    }
}
else
{
    rdone = TRUE;
    break;
}
}
while (!qdone)
{
    all_ok = TRUE;
    if (answ != '4')
    {
        cf = fgets(inline, 85, rdd);
        if (feof(rdd))
            qdone = TRUE;
        else
        {
            i = strlen(inline);
            inline[i-1] = '\0';
            substr(idrad, inline, 1, 8);
            substr(temp_str, inline, 10, 6);
            radius = atof(temp_str)/1000.0;
            inline[42] = '\0';
            printf("\nWorking core %s",inline);
        }
    }
    else
    {
        printf("Enter core id - (<ret> to quit): ");
        gets(idrad);
    }
}

```

Source-code of program AREA, continued.

```

if ((ci=strlen(idrad)) > 0 )
{
    for (i=ci; i<8; i++)
        idrad[i] = ' ';
    idrad[8] = '\0';
    printf("Enter radius: ");
    gets(temp_str);
    radius = atof(temp_str)/1000.0;
}
else
    qdone = TRUE;
}
if (!qdone)
{
    if (all_ok)
    {
        getrings(rwd, idrad, &first_yr, &last_yr,
                ringwidth, &all_ok, &year_cnt, &crit_yr, crit_val, elw);
        if (canopy != 0)
        {
            printf(" %4d",crit_yr);
            if (canopy == 1)
            {
                if (crit_yr == 0 || (canopy_year > 0 && crit_yr >= canopy_year))
                {
                    printf(" NOT USED\07");
                    all_ok = FALSE;
                }
            }
            else
            {
                if (canopy == 2 && crit_yr > 0)
                {
                    if (canopy_year == 0 || canopy_year >= crit_yr)
                    {
                        printf(" NOT USED\07");
                        all_ok = FALSE;
                    }
                }
            }
        }
    }
    if (all_ok)
    {
        b_area(year_cnt, ringwidth, radius, rbai, elw);
        splin(rbai, sbai, year_cnt, spline_len, &all_ok);
        if (all_ok)
        {
            ms=0.0;
            for (i=1; i <= year_cnt; i++)
                iindex[i] = rbai[i] / sbai[i];
            for (i=2; i < year_cnt; i++)
            {
                if ((iindex[i] == 0) && (iindex[i+1] == 0))
                    ms = ms+1;
                else
                    ms=ms + fabs(2*(iindex[i]-iindex[i-1])/(iindex[i]+iindex[i+1]));
            }
            ms=ms/(year_cnt-1);
            if (ind_file)
            {
                fprintf(bd, "\nRING WIDTHS %s MS=%7.3f %4d=%7.2f CM.\n",
                        idrad, ms, crit_yr, crit_val);
                if (elw < 2)
                    writef(bd, ringwidth, year_cnt+1, first_yr, last_yr);
                else
                    writef(bd, &ringwidth[1000], year_cnt+1, first_yr, last_yr);
                fprintf(bd, "\nRBAI %s\n", idrad);
                writef(bd, rbai, year_cnt+1, first_yr, last_yr);
                fprintf(bd, "\nSBAI %s\n", idrad);
            }
        }
    }
}

```

Source-code of program AREA, continued.

```

writef(bd, sbai, year_cnt+1, first_yr, last_yr);
fprintf(bd, "\nINDEX      %s\n", idrad);
writef(bd, iindex, year_cnt+1, first_yr, last_yr);
if (idx_file)
{
    writex(idx, iindex, cores, year_cnt+1, first_yr, last_yr, idrad);
}
fprintf(bd, "\nNORMAL INDEX %s\n", idrad);
normal(year_cnt, iindex, &ynf, &sdv);
writef(bd, iindex, year_cnt+1, first_yr, last_yr);
}
nyr = first_yr;
for (k=1; k <= year_cnt; k++)
{
    if (nyr >= syr && nyr <= eyr)
    {
        sub1 = nyr-syr+1;
        sum[sub1][1] = sum[sub1][1]+sbai[k];
        sum[sub1][2] = sum[sub1][2]+(sbai[k]*sbai[k]);
        sum[sub1][3] = sum[sub1][3]+1;
        summ[sub1][1] = sum[sub1][1]+rbai[k];
        summ[sub1][2] = summ[sub1][2]+(rbai[k]*rbai[k]);
    }
    nyr = nyr+1;
}
}
}
} /* end qdone */
ms = 0.0;
for (k=1; k <= n2; k++)
{
    if (sum[k][3] > 0)
    {
        means[k] = sum[k][1] / sum[k][3];
        average[k] = summ[k][1] / sum[k][3];
        iindex[k] = average[k] / means[k];
    }
    else
    {
        means[k] = 0;
        average[k] = 0;
        iindex[k] = 1;
    }
    if (sum[k][3] < 30.0)
    {
        tt = sum[k][3];
        ttable = ttable_vals[tt];
    }
    else
        ttable = 2.0;
    if (sum[k][3] <= 1.0)
        stdev[k] = 0.0;
    else
    {
        stdev[k] = sqrt((sum[k][2] - ((sum[k][1]*sum[k][1])/sum[k][3]))
            /((sum[k][3]-1)));
        stdev[k] = ttable/sqrt(sum[k][3])*stdev[k];
    }
    upper[k] = stdev[k] + means[k];
    lower[k] = means[k] - stdev[k];
    cores[k] = sum[k][3];
}
for (i=2; i<n2; i++)
{
    if ((iindex[i] == 0) && (iindex[i+1] == 0))
        ms = ms+1;
}

```

Source-code of program AREA, continued.

```

    else
        ms=ms + fabs(2*(iindex[i]-iindex[i-1])/(iindex[i]+iindex[i+1]));
    }
    ms=ms/(n2-1);
    fprintf(ixd1, "%s\n", title);
    writex(ixd1, iindex, cores, n2+1, syr, eyr, "CRNIDX");
    if (sta_file)
    {
        fprintf(sd, "\\NUMBER OF CORES:   MS=%7.3f\n", ms);
        writei(sd, cores, n2+1, syr, eyr);
        fprintf(sd, "\\NRBA1\n");
        writef(sd, average, n2+1, syr, eyr);
        fprintf(sd, "\\NSBA1\n");
        writef(sd, means, n2+1, syr, eyr);
        fprintf(sd, "\\NUPPER 95%% CONFIDENCE LIMIT:\n");
        writef(sd, upper, n2+1, syr, eyr);
        fprintf(sd, "\\NLOWER 95%% CONFIDENCE LIMIT:\n");
        writef(sd, lower, n2+1, syr, eyr);
        fprintf(sd, "\\NINDEX VALUE:\n");
        writef(sd, iindex, n2+1, syr, eyr);
        fprintf(sd, "\\NNORMALIZED INDEX:\n");
        normal(n2, iindex, &ynf, &sadv);
        writef(sd, iindex, n2+1, syr, eyr);
    }
    printf("\n");
    fclose(rwd);
    fclose(rdd);
    fclose(bd);
    fclose(sd);
    fclose(ixd);
    fclose(ixd1);
}

index(s, t)
char s[], t[];
{
    int i, j, k;
    for(i=0; s[i] != '\0'; i++)
    {
        for(j=i, k=0; t[k] != '\0' && s[j]==t[k]; j++, k++)
            if (t[k] == '\0') return(i);
    }
    return(-1);
}

cls()
{
    printf("\033[2J");
    fflush(stdout);
}

/*****
/* File name      : BAREA.C
*****/

b_area(years, rw, radius, a, elw)

int years;
double *rw;
double radius;
double *a;
int elw;

{
    int i;
    double rlen_prev, rlen_curr;
    rlen_prev = radius;

```

Source-code of program AREA, continued.

```

for (i=years-1; i>=0; i--)
{
    if (elw == 0)
    {
        rlen_curr = rlen_prev - rw[i+1];
        a[i+1] = 3.14159*((rlen_prev*rlen_prev) - (rlen_curr*rlen_curr));
        rlen_prev = rlen_curr;
    }
    else
    {
        if (elw == 1)
        {
            rlen_curr = rlen_prev - rw[i+1+1000];
            rlen_prev = rlen_curr;
            rlen_curr = rlen_prev - rw[i+1];
            a[i+1] = 3.14159*((rlen_prev*rlen_prev) - (rlen_curr*rlen_curr));
            rlen_prev = rlen_curr;
        }
        else
        {
            rlen_curr = rlen_prev - rw[i+1+1000];
            a[i+1] = 3.14159*((rlen_prev*rlen_prev) - (rlen_curr*rlen_curr));
            rlen_prev = rlen_curr;
            rlen_curr = rlen_prev - rw[i+1];
            rlen_prev = rlen_curr;
        }
    }
}

/*****
/* File name      : GETRADS.C
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

getrads(id, od, c_val, elw)

FILE *id, *od;
float c_val;
int elw;

{
    char idnumber[9], line[101];
    char c = ' ';
    char tempstr[9], *cf;
    char *fgets(), *substr();

    int i, j, value, years;
    int ci, first_year, last_year, crit_year;
    int done = FALSE, first = TRUE;
    int ci_val;
    unsigned int total;
    crit_year = 0;
    years = 0;
    ci_val = c_val*1000;
    while (!done)
    {
        cf = fgets(line, 100, id);
        line[80] = '\0';
        if (feof(id))
        {
            fprintf(od, "%s %6u %4d %4d %4d %4d\n",
                    idnumber, total, years, first_year, last_year, crit_year);
            done = TRUE;
        }
    }
}

```

Source-code of program AREA, continued.

```

        break;
    }
    substr(tempstr, line, 1, 8);
    if (strcmp(tempstr, idnumber) != 0)
    {
        if ( !first )
        {
            fprintf(od, "%s %6u %4d %4d %4d %4d\n",
                    idnumber, total, years, first_year, last_year, crit_year);
        }
        substr(idnumber, line, 1, 8);
        substr(tempstr, line, 9, 4);
        first_year = atoi(tempstr);
        last_year = first_year - 1;
        total = 0;
        crit_year = 0;
        years = 0;
        first = FALSE;
    }
    substr(tempstr, line, 12, 1);
    if ( !elw )
    {
        j = 10 - (atoi(tempstr) % 10);
    }
    else
    {
        j = (5 - (atoi(tempstr) % 5)) * 2;
        for (i = 1; i < j + 1; i++)
        {
            substr(tempstr, line, (7 + (i * 6)), 6);
            if (strcmp(tempstr, "99900") == 0)
                break;
            total = total + atoi(tempstr);
            if (elw > 0)
            {
                i++;
                substr(tempstr, line, (7 + (i * 6)), 6);
                total = total + atoi(tempstr);
            }
            last_year = last_year + 1;
            years = years + 1;
            if (crit_year == 0 && total >= ci_val)
                crit_year = last_year;
        }
    }
    /* end while */
}
/* end */

/*****
/* File name      : GETRINGS.C
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

getrings(rwd, idrad, first_yr, last_yr, rw, all_ok, year_cnt, cyr, cval, elw)

FILE *rwd;
char idrad[];
int *first_yr, *last_yr;

double *rw;
int *all_ok, *year_cnt, *cyr, elw;
double cval;

{
    int i, j, rend, rwdone=FALSE, first=TRUE;
    int ci, lyr, yrs, temp_yr, atoi();
    char *fgets();
    char check_end[7], idrw[9], c, *cf;
    char rwline[101], tempstr[10];

```

Source-code of program AREA, continued.

```

double atof(), cvt = 0.0;

strcpy(check_end, "99900");
*first_yr = 0;
lyr = 0;
yrs = 0;
*cyr = 0;
while (!rwdone)
{
    cf = fgets(rwline, 100, rwd); /* read ringwidth line */
    rwline[80] = '\0';
    if (feof(rwd)) /* EOF means id not found - ERROR */
    {
        *all_ok = FALSE;
        rwdone = TRUE;
        break;
    }
    else
    {
        substr(idrw, rwline, 1, 8);
        if ((rend = strcmp(idrw, idrad)) == 0) /* rw id */
        {
            if (*first_yr == 0)
            {
                substr(tempstr, rwline, 9, 4);
                *first_yr = atoi(tempstr); /* first year of core */
                lyr = atoi(tempstr);
                first = FALSE;
                if (elw)
                    i = 10 - (lyr%10);
                else
                    i = (5 - (lyr%5))*2;
            }
            else
            {
                i = (strlen(rwline)-13)/6;
                for (j = 1; j<=i; j++)
                {
                    substr(tempstr, rwline, 13+((j-1)*6), 6);
                    if (strcmp(tempstr, check_end) == 0)
                    {
                        lyr = lyr + yrs - 1;
                        *last_yr = lyr;
                        *year_cnt = yrs;
                        rwdone = TRUE;
                        break;
                    }
                    else
                    {
                        yrs = yrs + 1; /* number of years in core */
                        rw[lyrs] = atof(tempstr)/1000.0;
                        cvt=cvt+rw[lyrs];
                        if (elw != 0)
                        {
                            j++;
                            substr(tempstr, rwline, 13+((j-1)*6), 6);
                            rw[lyrs+1000] = atof(tempstr)/1000.0;

                            cvt=cvt+rw[lyrs];
                        }
                        if( *cyr == 0 && cvt >= cval)
                            *cyr=lyr-yrs-1;
                    }
                } /* end for */
            } /* end compare id */
            if (!first && rend != 0)
            {
                i= strlen(rwline);
                j=fseek(rwd, ((long)(i*(-1))), SEEK_CUR);
                rwdone = TRUE;
            }
        }
    }
}

```

Source-code of program AREA, continued.

```
        lyr = lyr + yrs - 1;
        *last_yr = lyr;
        *year_cnt = yrs;
        break;
    }
} /* end not EOF */
} /* end while */
} /* end getrings */

/*****
/* File name      : NORMAL.C
*****/

#include <math.h>

normal(n, x, xm, sd)
int n;
double *x, *xm, *sd;

{
    double an, sum, sumsq, var;
    double sqrt();
    int i;

    if (n <= 1)
    {
        puts("\nNumber of years is one or less. Program is terminating.\n\n");
        exit();
    }
    an = (double)n;
    sum = 0.0;
    sumsq = 0.0;
    for (i=1; i<=n; i++)
    {
        sum = sum+x[i];
        sumsq=sumsq+((x[i])*(x[i]));
    }
    *xm = sum/an;
    var = (sumsq-an*((*xm)*(*xm)))/(an-1.0);
    if (var < 0.0)
        var = var*(-1.0);
    *sd = sqrt((double)var);
    if (*sd == 0.0)
        return;
    for (i=1; i<=n; i++)
        x[i] = (x[i]-(*xm))/(*sd);
    return;
}

/*****
/* File name      : SPLINE.C
*****/

#define TRUE 1
#define FALSE 0

splin(y, spl, n, spline, all_ok)

double *y, *spl;
int n;
double spline;
int *all_ok;

{
    int i, j, k, nc, ncp1, nm2, m, i1, i2, l, jm1;
    int imncp1, nc1, iw, kl, n1, k1;
```

Source-code of program AREA, continued.

```

double a[1000][3], b[1000];
double d1, d2, rn, tot, p;
double pct = 0.50000000;
double pi = 3.1415926535897935;
double p1,p2,p3,p4,p5;
double c1[5], c2[4];
double cos(), sqrt(), fabs();

c1[1]=1.000000;
c1[2]=-4.000000;
c1[3]=6.000000;
c1[4]=-2.000000;
c2[1]=0.000000;
c2[2]=0.3333333333333333;
c2[3]=1.3333333333333333;

if ( n > 1800 || n < 4)
{
    if (n > 1800)
    {
        printf("\n%cSpline capacity (1800) exceeded. \n(7/7)");
        printf("    Program terminating.\n");
        all_ok = FALSE;
        return;
    }
}
rm2 = n-2;
p= ((1.0/(1.0-pct)-1.0)*6.0*((cos(pi*2.0/spline)-1.0)*(cos(pi*2.0/spline)-1.0)))
/(cos(pi*2.0/spline)+2.0);
for (i=1; i<=rm2; i++)
{
    for (j=1; j<4; j++)
    {
        a[i][j] = c1[j]+p*c2[j];
        b[i] = y[i] + c1[4] * y[i+1] + y[i+2];
    }
}
a[1][1] = c2[1];
a[1][2] = c2[1];
a[2][1] = c2[1];
nc = 2;
/* Begin LUDAPB */
rn = 1.0/(rm2*16.0);
d1 = 1.0;
d2 = 0.0;
ncp1 = nc + 1;
for (i=1; i<=rm2; i++)
{
    imncp1 = i-ncp1;
    if ( 1-imncp1 > 1)
        i1 = 1-imncp1;
    else
        i1 = 1;
    for (j=i1; j<=ncp1; j++)
    {
        l = imncp1 + j;
        i2 = ncp1 - j;
        tot = a[i][j];
        jm1 = j - 1;
        if (jm1 > 0)
        {
            for (k=1; k<=jm1; k++)
            {
                m = i2 + k;
                tot = tot - (a[i][k] * a[l][m]);
            }
        }
    }
}

```

Source-code of program AREA, continued.

```

if (j == ncp1)
{
    if (a[i][j] + tot * rn <= a[i][j])
    {
        spl[1] = 9999.0;
        printf("%c%cSpline matrix not positive definite.\n\7\7");
        printf(" Program is terminating.\n");
        all_ok = FALSE;
        return;
    }
    else
    {
        a[i][j] = 1.0 / sqrt(tot);
        /* Update the determinant */
        d1 = d1 * tot;
        if (fabs(d1) > 1.00000)
        {
            while (fabs(d1) > 1.000)
            {
                d1 = d1 * 0.062500;
                d2 = d2 + 4.0;
            }
        }
        if (fabs(d1) <= 0.062500)
        {
            while (fabs(d1) <= 0.062500)
            {
                d1 = d1 * 16.0;
                d2 = d2 - 4.0;
            }
        }
        a[i][j] = tot * a[l][ncp1];
    }
}
}
else
{
    a[i][j] = tot * a[l][ncp1];
}
}
}
/* End LUDAPS / Begin LUELPS */
/* Solution LY = B */
if (all_ok)
{
    ncl = nc + 1;
    iw = 0;
    l = 0;
    for (i=1; i <= nm2; i++)
    {
        tot = b[i];
        if (nc > 0)
        {
            if (iw != 0)
            {
                l = l + 1;
                if (l > nc)
                {
                    l = nc;
                    k = ncl - l;
                    kl = i - l;
                    for (j=k; j <= nc; j++)
                    {
                        tot = tot - (b[kl] * a[i][j]);
                        kl = kl + 1;
                    }
                }
            }
            else
            {
                if (tot != 0.0)
                    iw = 1;
            }
        }
    }
}

```

Source-code of program AREA, continued.

```

        b[i] = tot * a[i][nc1];
    }
}
/* Solution UX = Y */
if (all_ok)
{
    b[rm2] = b[rm2] * a[rm2][nc1];
    n1 = rm2 + 1;
    for (i=2; i <= rm2; i++)
    {
        k = n1 - i;
        tot = b[k];
        if (nc > 0.0)
        {
            k1 = k + 1;
            if (k+nc > rm2)
                k1 = rm2;
            else
                k1 = k+nc;
            l = 1;
            for (j=k1; j<=k1; j++)
            {
                tot = tot - (b[j] * a[j][nc1-l]);
                l = l + 1;
            }
        }
        b[k] = tot * a[k][nc1];
    }
}
/* end LUELPB */
if (all_ok)
{
    for (i = 3; i <= rm2; i++)
        spl[i] = b[i-2] + c1[4] * b[i-1] + b[i];
    spl[1] = b[1];
    spl[2] = c1[4] * b[1] + b[2];
    spl[n-1] = b[rm2-1] + c1[4] * b[rm2];
    spl[n] = b[rm2];
    for (i=1; i <= n; i++)
        spl[i] = y[i] - spl[i];
}
}

/*****
/* File name      : WRT_FLS.C
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

writef(fp, prtarr, j, fyr1, ldec1)
int j, fyr1, ldec1;
FILE *fp;
double prtarr[];

{
    int l, i, jb, je;
    int done = FALSE;
    prtarr[j] = 99.99;
    jb = 1;
    je = (10 - (fyr1 % 10));
    while (!done)
    {
        if (i == j)
        {
            done = TRUE;
            break;

```

Source-code of program AREA, continued.

```
    }
    fprintf(fp, "%4d", fyr1);
    for (i=jb; i <= je; i++)
    {
        if (i == j)
        {
            done = TRUE;
            break;
        }
        fprintf(fp, "%7.3f", prtarr[i]);
    }
    fprintf(fp, "\n");
    if (i <= 10)
        fyr1 = fyr1 + i - 1;
    else
        fyr1 = fyr1 + 10;
    jb = je + 1;
    je = je + 10;
}

writei(fp, prtarr, j, fyr1, ldec1)
int j, fyr1, ldec1;
FILE *fp;
int prtarr[];

{
    int l, i, jb, je, temp;
    int done = FALSE;
    prtarr[j] = 9999;
    jb = 1;
    je = (10 - (fyr1 % 10));
    while (!done)
    {
        if (i == j)
        {
            done = TRUE;
            break;
        }
        fprintf(fp, "%4d", fyr1);
        for (i=jb; i <= je; i++)
        {
            if (i == j)
            {
                done = TRUE;
                break;
            }
            temp = prtarr[i];
            fprintf(fp, "%7d", temp);
        }
        fprintf(fp, "\n");
        if (i <= 10)
            fyr1 = fyr1 + i - 1;
        else
            fyr1 = fyr1 + 10;
        jb = je + 1;
        je = je + 10;
    }
}

wrtidx (fp, prtarr, cor, j, fyr, lyr, id)

FILE *fp;
double prtarr[];
int cor[], j, fyr, lyr;
char id[];

{
    int l, f, jb, je, fdec, ldec, temp;
```

Source-code of program AREA, continued.

```
int done = FALSE;

fdec = (fyr/10)*10;
ldec = (lyr/10)*10;
id[6] = '\0';
jb = 1;
je = fyr % 10;
if (je != 0)
{
    fprintf(fp, "%s%4d", id, fyr);
    for (i=jb; i<=je; i++)
        fprintf(fp, "9990 0");
}
for (i=1; i<=j; i++)
{
    if(((fyr+i-1)%10) == 0)
        fprintf(fp, "%s%4d", id, fyr+i-1);
    tempi = ptarr[i]*1000;
    fprintf(fp, "%4d%3d", tempi, cor[i]);
    if (((fyr+i-1)%10) == 9)
        fprintf(fp, "\n");
    if (i == j-1)
        break;
}
je = 10-(lyr%10);
if (je != 0)
    for (i=1; i<=je; i++)
        fprintf(fp, "9990 0");
fprintf(fp, "\n");
}
```

Source-code listing for program PLOT.

```

/*****
/* Program name      : PLOT.C
/* Purpose           : Menu-driven program to plot tree-ring data.
/* Input             : Data file created by program AREA.
/* Output            : To HP 7475A plotter.
/* Programmer        : Michael L. Field
/* Created for       : Tree-Ring Laboratory
/*                   : Mail Stop #461
/*                   : U.S. Geological Survey
/*                   : Reston, Virginia 22092
/* Files used        : CHNG_ALL.C
/*                   : FIL_ARRS.C
/*                   : LIST_ALL.C
/*                   : PLT_UTIL.C
/*                   : PRINT_ALL.C
/*                   : P_PLOT.C
/*                   : READ_FLS.C
/*                   : S_PLOT.C
/*                   : WRITEF.C
/*                   : COMMON.C
/* Date Created      :
/* Modifications     :
/* Purpose           :
/* Date              :
/* Programmer        :
*****/

#include <stdio.h>
#include <graph.h>
#include <string.h>
#include <math.h>
#include <bios.h>
#include <stdlib.h>
#include "plot_ver.h"

/* Functions - prototypes and macros */

int menu(int, int, char* []); /* Menus */
void box(int, int, int, int);
void itemize(int, int, char*, int);

/* Control pen and color */

unsigned cursor(unsigned);

/* Constants */

#define PI 3.141593
#define DEFAULT -1
#define TRUE 1
#define FALSE 0
#define NULL 0
#define TCUSROFF 0x2020
#define LASTATR 15
#define NLASTATR 14

/* Scan codes */

#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
#define ENTER 28

/* Structures for configuration and other data */

struct videoconfig vc;

```

Source-code listing for program PLOT, continued.

```

/* Initial and work arrays for remappable colors. */

long iColors[] = (
    _BLACK,      _BLUE,      _GREEN,      _CYAN,
    _RED,        _MAGENTA,    _BROWN,    _WHITE,
    _GRAY,       _LIGHTBLUE,  _LIGHTGREEN, _LIGHTCYAN,
    _LIGHTRED,   _LIGHTMAGENTA, _LIGHTYELLOW, _BRIGHTWHITE );

long wColors[256];

/* Array and enum for main menu */

char *mnuMain[] = (
    "1. Read Chronology(.CRN) file.",
    "2. Read Individual(.IND) file.",
    "3. Read Tektronics data file.",
    "4. Read Other Data(.DAT) file.",
    "5. Input datum from keyboard.",
    "6. Directory listing.",
    "7. Set up plotter port.",
    "8. Quit.",
    NULL );

enum ( READS, READI, READT, READD, KYBD, DIRT, PORT, QUIT );

/* Array for plot menu */

char *mnuPlot[] = (
    "1. Change title.",
    "2. Change labels, axis or window values.",
    "3. Plot and graph routines.",
    "4. List data, status info, or ascii file.",
    "5. Save keyboard file to disk.",
    "6. Main menu.",
    "7. Quit.",
    NULL );

char *mnuDevice[] = (
    "1. Screen Plot.",
    "2. HP-7475 8-1/2 x 11.",
    "3. HP-7475 11 x 14.",
    "4. Prev menu.",
    NULL );

char *mnuChng[] = (
    "1. Change X label.",
    "2. Change Y label.",
    "3. Change X axis.",
    "4. Change Y axis.",
    "5. Change X data window.",
    "6. Change Y data window.",
    "7. Prev menu.",
    NULL );

char *mnuGraph[] = (
    "1. Plot BAI.",
    "2. Plot Ring widths.",
    "3. Plot Indices.",
    "4. Prev menu.",
    NULL );

char *mnuIndtyp[] = (
    "1. Standard index.",
    "2. Normalized index.",
    "3. Previous menu.",
    NULL );

```

Source-code listing for program PLOT, continued.

```
char *mnuLmain[] = (  
    "1. List data.",  
    "2. List status.",  
    "3. List ascii file.",  
    "4. Prev menu.",  
    NULL );
```

```
char *mnuLdata[] = (  
    "1. Screen.",  
    "2. Printer.",  
    "3. Prev menu.",  
    NULL );
```

```
char *mnuPort[] = (  
    "1. Port.",  
    "2. # Characters.",  
    "3. Stop bits.",  
    "4. Parity.",  
    "5. Baud rate.",  
    "6. Main menu.",  
    NULL );
```

```
char *mnuPport[] = (  
    "1. Com1.",  
    "2. Com2.",  
    "3. Prev menu.",  
    "4. Main menu.",  
    NULL );
```

```
char *mnuChar[] = (  
    "1. 7 characters.",  
    "2. 8 CHARACTERS.",  
    "3. Prev menu.",  
    "4. Main menu.",  
    NULL );
```

```
char *mnuStop[] = (  
    "1. 1 stop bit.",  
    "2. 2 stop bits.",  
    "3. Prev menu.",  
    "4. Main menu.",  
    NULL );
```

```
char *mnuParity[] = (  
    "1. None.",  
    "2. Odd.",  
    "3. Even.",  
    "4. Prev menu.",  
    "5. Main menu.",  
    NULL );
```

```
char *mnuBaud[] = (  
    "1. 300",  
    "2. 600",  
    "3. 1200",  
    "4. 2400",  
    "5. 4800",  
    "6. 9600",  
    "7. Prev menu.",  
    "8. Main menu.",  
    NULL );
```

/* Arrays for video modes */

```
char *mnuModes[] = (  
    "MRES4COLOR",  
    "MRESNOCOLOR",  
    "MRESBW",  
    "MRES16COLOR",
```

Source-code listing for program PLOT, continued.

```

        "HRES16COLOR",
        "ERESCOLOR",
        "VRES2COLOR",
        "VRES16COLOR",
        "HRES256COLOR",
        NULL );

/* Array for modes */

int modes[] = (
    _MRES4COLOR,
    _MRESMOCOLOR,
    _HRESBW,
    _MRES16COLOR,
    _HRES16COLOR,
    _ERESCOLOR,
    _VRES2COLOR,
    _VRES16COLOR,
    _MRES256COLOR,
    _ERESMOCOLOR,
    DEFAULT );

/* Structure for menu attributes (variables for color and monochrome) */

struct mnuAtr {
    int    fgNormal, fgSelect, fgBorder;
    long   bgNormal, bgSelect, bgBorder;
    int    centered;
    char   nw[2], ne[2], se[2], sw[2], ns[2], ew[2];
} menus = (
    0, 15, 4,
    3, 4, 3,
    TRUE,
    "\xda", "\xbf", "\xd9", "\xc0", "\xb3", "\xc4"
);

struct mnuAtr bmenus = (
    0x70, 0xf, 0x70,
    0x7, 0x70, 0x7,
    TRUE,
    "\xda", "\xbf", "\xd9", "\xc0", "\xb3", "\xc4"
);

char mess1[] = ( "Basal Area plotting routines" );
char mess2[] = (
    "Move to menu selection with cursor keys, press ENTER to select" );
char mess3[] = ( "(Press enter key to continue)\n" );
char mess4[] = ( "File successfully read.\n" );
char mess5[] = ( "\nEnter Chronology file name: " );
char mess6[] = ( "\nEnter Individual file name: " );
char mess6a[] = ( "\nEnter Individual file name (default=: " );
char mess6b[] = ( "): " );
char mess7[] = ( "\nEnter TEKTRONICS file name: " );
char mess9[] = ( "\nEnter Other Data file name: " );
char mess8[] = ( "\nPROBLEM READING FILE.\n" );
char mess10[] = ( "\n                File successfully written." );

int lmess1 = sizeof(mess1), lmess2 = sizeof(mess2), lmess3 = sizeof(mess3),
    lmess4 = sizeof(mess4);

short style[16] = {0x1, 0x3, 0x7, 0xf, 0x1f, 0x3f, 0x7f, 0xff, 0x1ff, 0x3ff,
    0x7ff, 0xfff, 0x1fff, 0x3fff, 0x7fff, 0xffff};

/* COM PORT SETUP */

static int port;
static int rxtx;
static int stop;
static int parity;
static int baud;

```

Source-code listing for program PLOT, continued.

```

main ()
{
    static char title[81], rsname[41], temp_ind[41];
    static char x_lab[41] = "YEAR";
    static char y_lab[41] = "BASAL AREA INCREMENT";
    static char core[9];
    static int t1, t2, t3, x1, x2, y1, y2, sc, cxy, lmu, pmu;
    static int years, first_yr, last_yr, lpt;
    static int all_ok = TRUE, Pdone, Mdone, Ldone, Bdone, Cdone;
    static int Ddone, Ldone, Lmdone;
    static int i, j, k, l, m, line_cnt, cyr;
    static int first;
    static int temp;
    static int p1, p2;
    static int w1, w2, w3, w4, q1, q2, q3;
    static int chngd_x1_x2 = FALSE, chngd_y1_y2 = FALSE;
    static int chngd_w1_w2 = FALSE, chngd_w3_w4 = FALSE;
    static float ys, xs, xadd, yadd, ysacle, xsacle;
    static double y_min, y_max;
    static double rings[750], rbai[750], sbai[750];
    static double up[750], low[750], index[750];
    static double nindex[750];
    static double q4, ms, cval;

    char *gets(), *p;
    char c;
    char c4;
    char temp[5];
    char temp_str[8];
    char temp_dir[41];
    char dirnam[128];
    char choice3[2], choice4[2];

    int choice, choice2, crow, ccol;
    int tmode, vmode, done;
    long bcolor;

    FILE *rs;
    FILE *setupinf(), *setupout();

    int idumy1, idumy2, idumy3, idumy4, indtyp;
    double ddumy1, ddumy2, ddumy3;
    char cdumy1[81], cdumy2[9];

    indtyp = 0;
    port = 0;
    rxtx = 3;
    stop = 0;
    parity = 24;
    baud = 224;
    _bios_serialcom(_COM_INIT, port, (rxtx | stop | parity | baud));
    _getvideoconfig(&vc);
    crow = vc.numtextrows / 2;
    ccol = vc.numtextcols / 2;

    /* Select best text and graphics modes and adjust menus

switch (vc.adapter)
{
    case _MDPA :
        puts("No graphics mode available.\n");
        exit(0);
    case _CGA :
        mnuModes[3] = NULL;
        vmode = _HRESBW;
        break;
    case _EGA :
        mnuModes[6] = NULL;

```

Source-code listing for program PLOT, continued.

```

        if (vc.memory > 64)
            vmode = _ERESCOLOR;
        else
            vmode = _HRES16COLOR;
        break;
    case _VGA :
        vmode = _VRES16COLOR;
        break;
    case _MCGA :
        vmode = _MRES256COLOR;
        break;
    }
    switch (vc.mode)
    {
    case _TEXTBW80 :
        menus = bwmenus;
    case _TEXTBW40 :
        _setvideomode(_TEXTBW80);
        break;
    case _TEXTC40 :
        tmode = _TEXTC80;
        break;
    case _TEXTMONO :
    case _ERESNOCOLOR :
        menus = bwmenus;
        tmode = _TEXTC80;
        vmode = _ERESNOCOLOR;
        mnuMain[7] = NULL;
        break;
    default:
        tmode = _TEXTC80;
    }

    _setvideomode(tmode);
    _settextposition(7,40 - (lmess1 / 2));
    _outtext(mess1);
    _settextposition(9,40-(sizeof(version)/2));
    _outtext(version);
    _settextposition(11,40 - (lmess3 / 2));
    _outtext(mess3);
    gets(temp_str);
    cls();
    _settextposition(2,40 - (lmess2 / 2));
    _outtext(mess2);
    /* Select and branch to menu choices */
    for (;;)
    {
        Mdone = FALSE;
        while (!Mdone)
        {
            chngd_x1_x2 = FALSE;
            chngd_y1_y2 = FALSE;
            chngd_w1_w2 = FALSE;
            chngd_w3_w4 = FALSE;
            temp_ind[0] = '\0';
            choice = menu(crow,ccol,mnuMain);
            if (vmode == 6 || vmode == 14 || vmode == 16 || vmode == 18)
                xs = 2;
            else
                xs = 1;
            if (vmode == 15 || vmode == 16)
                ys = 1.75;
            else
            {
                if (vmode == 17)
                    ys = 2.4;
                else
                    if (vmode == 18)
                        ys = 2.0;
            }
        }
    }

```

Source-code listing for program PLOT, continued.

```

        else
            ys = 1;
    }
    _setvideomode (_DEFAULTMODE);
    switch (choice)
    {
        case QUIT :
            exit(0);
            break;
        case READS : /* read chron file */
            all_ok = TRUE;
            _setvideomode (_DEFAULTMODE);
            _outtext(mess5);
            gets(rsname);
            if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
            {
                read_crn(rs, rings, rbai, sbai, up, low, index, nindex, &years,
                    &first_yr, &last_yr, &y_max, title, &all_ok, &ms);
                fclose(rs);
                if (all_ok)
                {
                    tempi = strlen(rsname);
                    for (i=tempi; i>=0; i--)
                    {
                        if (rsname[i] == '.')
                        {
                            substr(temp_ind, rsname, 1, i+1);
                            strcat(temp_ind, "ind");
                            break;
                        }
                    }
                    cls();
                    _settextposition(10,40 - (lmess4 / 2));
                    _outtext (mess4);
                    _settextposition(11,40 - (lmess3 / 2));
                    _outtext(mess3);
                    gets(temp_str);
                    Mdone = TRUE;
                }
            }
            else
            {
                puts("\a");
                cls();
                _outtext (mess8);
                _outtext ("    File may be wrong type.");
                _settextposition(11,40 - (lmess3 / 2));
                _outtext(mess3);
                gets(temp_str);
            }
    }
    else
    {
        puts("\a");
        cls();
        cntinue();
    }
    break;
    case READI : /* read individual file */
        all_ok = TRUE;
        _setvideomode (_DEFAULTMODE);
        _outtext(mess6);
        gets(rsname);
        if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
        {
            read_ind(rs, up, rbai, sbai, index, nindex, &years,
                &first_yr, &last_yr, &y_max, title,
                &ms, &cyr, &cval, &all_ok, core);
            fclose(rs);
        }
    }
}

```

Source-code listing for program PLOT, continued.

```

if (all_ok)
{
    strcpy(temp_ind, rsname);
    cls();
    _outtext (mess4);
    _settextposition(11,40 - (lmess3 / 2));
    _outtext(mess3);
    gets(temp_str);
    Mdone = TRUE;
}
else
{
    puts("\a");
    _outtext(mess8);
    _outtext("    End of file reached before finding core.\n");
    _outtext("    File may be wrong type or\n");
    _outtext("    core may not be in file.\n");
    _outtext(mess3);
    gets(temp_str);
}
}
else
{
    puts("\a");
    cls();
    cntnue();
}
break;
case READT : /* read tektronics file */
    _setvideomode (_DEFAULTMODE);
    _outtext(mess7);
    gets(rsname);
    if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
    {
        read_tek(rs, rbai, &years, &first_yr, &last_yr, &y_max, title);
        fclose(rs);
        cls();
        _outtext (mess4);
        _settextposition(11,40 - (lmess3 / 2));
        _outtext(mess3);
        gets(temp_str);
        Mdone = TRUE;
    }
    else
    {
        puts("\a");
        cls();
        cntnue();
    }
    break;
case READO : /* read other data file */
    all_ok = TRUE;
    _setvideomode (_DEFAULTMODE);
    _outtext(mess9);
    gets(rsname);
    if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
    {
        read_oth(rs, rbai, &years, &first_yr, &last_yr,
                &y_max, title, &all_ok);
        fclose(rs);
        if (all_ok)
        {
            cls();
            _outtext (mess4);
            _settextposition(11,40 - (lmess3 / 2));
            _outtext(mess3);
            gets(temp_str);
            cntnue();
        }
    }
}

```

Source-code listing for program PLOT, continued.

```

        Mdone = TRUE;
    }
    else
    {
        puts("\a");
        cls();
        _outtext (mess8);
        _outtext ("    File may be wrong type.");
        cntnue();
    }
}
else
{
    puts("\a");
    cls();
    cntnue();
}
break;
case KYBD : /* get data from keyboard */
    _setvideomode (_DEFAULTMODE);
    puts("Enter title: ");
    gets(title);
    cls();
    puts("Enter first year: ");
    gets(temp_str);
    first_yr = atoi(temp_str);
    last_yr = first_yr-1;
    years = 0;
    y_min = 0.0;
    y_max = 0.0;
    cls();
    done = FALSE;
    puts("\n\n                Enter QUIT to terminate values.\n");
    while(!done)
    {
        printf("%%d = ",last_yr+1);
        gets(temp_str);
        if (temp_str[0] == 'Q' || temp_str[0] == 'q')
            done = TRUE;
        else
        {
            years = years+1;
            last_yr = last_yr+1;
            rbel[years] = atof(temp_str);
            if (rbel[years] > y_max)
                y_max = rbel[years];
        }
    }
    cls();
    cntnue();
    Mdone = TRUE;
    break;
case DTRL : /* list directory */
    _setvideomode (_DEFAULTMODE);
    printf("\n\n\nEnter full pathname (enter=current):");
    gets(temp_dir);
    cls();
    strcpy(dirmam, "dir ");
    if ((l = strlen(temp_dir)) != 0)
        strcat(dirmam, temp_dir);
    strcat(dirmam, " /p");
    i = system(dirmam);
    cntnue();
    gets(temp_str);
    break;
case PORT : /* set up plotter port */
    _setvideomode (tmode);
    cls();
    _outtext("\nRoutine not completed yet\n");

```

Source-code listing for program PLOT, continued.

```
continue();
Cdone = FALSE;
while (!Cdone)
{
    cls();
    p1 = menu(crow, ccol, mnuPort);
    switch(p1)
    {
        case 5 :
            Cdone = TRUE;
            break;
        case 0 :
            Bdone = FALSE;
            while(!Bdone)
            {
                cls();
                p2 = menu(crow, ccol, mnuPort);
                switch(p2)
                {
                    case 3 :
                        Bdone = TRUE;
                        Cdone = TRUE;
                        break;
                    case 0 :
                        port = 0;
                        break;
                    case 1 :
                        port = 1;
                        break;
                    case 2 :
                        Bdone = TRUE;
                        break;
                    default :
                        break;
                }
            }
            break;
        case 1 :
            Bdone = FALSE;
            while(!Bdone)
            {
                cls();
                p2 = menu(crow, ccol, mnuChar);
                switch(p2)
                {
                    case 3 :
                        Bdone = TRUE;
                        Cdone = TRUE;
                        break;
                    case 0 :
                        rxtx = _COM_CHR7;
                        break;
                    case 1 :
                        rxtx = _COM_CHR8;
                        break;
                    case 2 :
                        Bdone = TRUE;
                        break;
                }
            }
            break;
        case 2 :
            Bdone = FALSE;
            while(!Bdone)
            {
                cls();
                p2 = menu(crow, ccol, mnuStop);
                switch(p2)
                {
```

Source-code listing for program PLOT, continued.

```

        case 3 :
            Bdone = TRUE;
            Cdone = TRUE;
            break;
        case 0 :
            stop = _COM_STOP1;
            break;
        case 1 :
            stop = _COM_STOP2;
            break;
        case 2 :
            Bdone = TRUE;
            break;
        default :
            break;
    )
)
break;
case 3 :
    Bdone = FALSE;
    while(!Bdone)
    {
        cls();
        p2 = menu(crow, ccol, mnuParity);
        switch(p2)
        {
            case 4 :
                Bdone = TRUE;
                Cdone = TRUE;
                break;
            case 0 :
                parity = _COM_NOPARITY;
                break;
            case 1 :
                parity = _COM_ODDPARITY;
                break;
            case 2 :
                parity = _COM_EVENPARITY;
                break;
            case 3 :
                Bdone = TRUE;
                break;
            default :
                break;
        }
    }
    break;
case 4 :
    Bdone = FALSE;
    while (!Bdone)
    {
        cls();
        p2 = menu(crow, ccol, mnuBaud);
        switch(p2)
        {
            case 7 :
                Bdone = TRUE;
                Cdone = TRUE;
                break;
            case 0 :
                baud = _COM_300;
                break;
            case 1 :
                baud = _COM_600;
                break;
            case 2 :
                baud = _COM_1200;
                break;
        }
    }
}

```

Source-code listing for program PLOT, continued.

```

        case 3 :
            baud = _COM_2400;
            break;
        case 4 :
            baud = _COM_4800;
            break;
        case 5 :
            baud = _COM_9600;
            break;
        case 6 :
            Bdone = TRUE;
            break;
        default :
            break;
    }
}
break;
}
_bios_serialcom(_COM_INIT, port, (rxtx | stop | parity | baud));
}
default :
    break;
} /* end switch choice */
cls();
} /* end while Bdone */
indtyp = 0;
y1 = 0;
y_ticks(&t2, y_max, &y2);
x_ticks(&t1, &x1, &x2, years, first_yr);
q1 = x1;
q2 = x2;
q3 = y2;
w1 = x1;
w2 = x2;
w3 = y1;
w4 = y2;
idumy1 = first_yr;
idumy4 = years;
list_inf(title, x_lab, y_lab, first_yr, last_yr, y_max, q1, q2, q3,
        x1, x2, y2, w1, w2, w3, w4, ms, cyr, cval, choice);
gets(temp);
cls();
_settextposition(2,40 - (lmess2 / 2));
_outtext(mess2);
Pdone = FALSE;
i = strlen(title);
title[i+1] = '\0';
while (!Pdone)
{
    choice2 = menu(crow,ccol,mnuPlot);
    _setvideomode (_DEFAULTMODE);
    switch (choice2)
    {
        case 6 : /* QUIT */
            exit(0);
        case 0 : /* CHNG_T */
            chng_t(title);
            break;
        case 1 : /* CHNG */
            Cdone = FALSE;
            while (!Cdone)
            {
                cls();
                cxy = menu(crow,ccol,mnuChng);
                cxy = cxy*10;
                switch(cxy)
                {

```

Source-code listing for program PLOT, continued.

```

    case 16 :
        Cdone = TRUE;
        break;
    case 10 : /* change x label */
        chng_x(x_lab);
        break;
    case 11 : /* change y label */
        chng_y(y_lab);
        break;
    case 12 : /* change x axis values */
        chngd_x1_x2 = TRUE;
        chng_x_vals(&x1, &x2, &t1);
        break;
    case 13 : /* change y axis values */
        chngd_y1_y2 = TRUE;
        chng_y_vals(&y1, &y2, &t2);
        break;
    case 14 : /* change x window */
        chngd_w1_w2 = TRUE;
        chng_x_win(&w1, &w2, &x1, &x2);
        break;
    case 15 : /* change y window */
        chngd_w3_w4 = TRUE;
        chng_y_win(&w3, &w4, &y1, &y2);
        break;
    default :
        break;
    }
}
break;
case 2 : /* plot and graph routines */
    Ddone = FALSE;
    while (!Ddone)
    {
        cls();
        pmnu = menu(crow, ccol, mnuPgraph);
        pmnu = pmnu+20;
        switch(pmnu)
        {
            case 23 : /* prev menu */
                Ddone = TRUE;
                break;
            case 20 : /* bai */
                Cdone = FALSE;
                while (!Cdone)
                {
                    _setvideomode(tmode);
                    sc = menu(crow, ccol, mnuDevice);
                    first = TRUE;
                    if (sc == 3)
                    {
                        plot_stp(x2, y2, &port);
                        Cdone = TRUE;
                        break;
                    }
                }
                if (!chngd_y1_y2)
                {
                    y_tics(&t2, y_max, &y2);
                    y1 = y_min;
                }
                if (!chngd_w3_w4)
                {
                    w3 = y1;
                    w4 = y2;
                }
                if (!chngd_x1_x2)
                    x_tics(&t1, &x1, &x2, idumy4, idumy1);
            }
        }
    }

```

Source-code listing for program PLOT, continued.

```

if (lchngd_w1_w2)
(
w1 = x1;
w2 = x2;
)
setvideomode (vmode);
if (sc == 0)
(
yscale = (16.0*ys)/t2;
xscale = (40.0*xs)/t1;
settextcolor(_RED);
g_labs(title, x_lab, y_lab, x1, y2, t1, t2, 2, 0);
settextcolor(_WHITE);
g_tics(xs, ys, 4.0, 28, 16.0, 7, w1, w2, w3, w4,
xscale, yscale, x1, x2, y1, y2, t1, t2);
g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
j = 1;
)
else
(
/* DO PLOT */
setvideomode (_DEFAULTMODE);
cls();
p_plot(title, x_lab, y_lab, x1, x2, y1, y2, t1, t2, 4, FALSE,
w1, w2, w3, w4, sc, &port, FALSE);
setvideomode (_DEFAULTMODE);
)
p = itoa(choice, choice3, 10);
c4 = ' ';
while (c4 != 'P' && c4 != 'p')
(
settextcolor(_RED);
switch (choice3[0])
(
case '0' :
outtext("(R)bai (S)bai (U)pper (L)ow\n");
break;
case '1' :
outtext("(R)bai (S)bai\n");
break;
case '2' :
case '3' :
case '4' :
c4 = 'X';
outtext("\n");
break;
default :
break;
) /* end switch choice3 */
settextcolor(_WHITE);
if (choice3[0] == '2' || choice3[0] == '3' || choice3[0] == '4')
(
if (!first)
c4 = getch();
)
else
c4 = getch();
switch(c4)
(
case 'R' :
case 'r' :
case 'X' :
if (c4 == 'X')
first = FALSE;
if (sc == 0)
g_line(j,xadd, yadd, xscale, yscale, idumy4,
rbai, vmode, 1);

```

Source-code listing for program PLOT, continued.

```

else
(
/* DO PLOT */
plt_line(rbai, idumy1, idumy4, x2, y2, 1.0, &port);
cls();
)
break;
case 'S' :
case 's' :
if (sc == 0)
g_line(j, xadd, yadd, xscale, yscale, idumy4,
sbai, vmode, 1);
else
(
/* DO PLOT */
plt_line(sbai, idumy1, idumy4, x2, y2, 1.0, &port);
cls();
)
break;
case 'U' :
case 'u' :
if (sc == 0)
g_line(j, xadd, yadd, xscale, yscale, idumy4,
up, vmode, 1);
else
(
/* DO PLOT */
plt_line(up, idumy1, idumy4, x2, y2, 1.0, &port);
cls();
)
break;
case 'L' :
case 'l' :
if (sc == 0)
g_line(j, xadd, yadd, xscale, yscale, idumy4,
low, vmode, 1);
else
(
/* DO PLOT */
plt_line(low, idumy1, idumy4, x2, y2, 1.0, &port);
cls();
)
break;
/* DO PLOT */
case 'C' :
case 'c' :
choice3[0] = '0';
c4 = '0';
_outtext(mess5);
gets(rsname);
if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
(
all_ok = TRUE;
choice = READS;
chngd_x1_x2 = TRUE;
chngd_y1_y2 = TRUE;
chngd_w1_w2 = TRUE;
chngd_w3_w4 = TRUE;
read_crn(rs, rings, rbai, sbai, up, low, index,
nindex, &idumy4, &idumy1, &idumy2, &ddumy1,
cdumy1, &all_ok, &ddumy2);
fclose(rs);
if (all_ok)
(
if (sc == 0)
g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
)
)

```

Source-code listing for program PLOT, continued.

```

        else
        {
            puts("\a");
            _outtext(mess8);
            gets(temp);
        }
    }
    else
    {
        puts("\a");
        gets(temp);
    }
    break;
case 'I' :
case 'i' :
    choice3[0] = '1';
    _outtext(mess6a);
    _outtext(temp_ind);
    _outtext(mess6b);
    gets(rsname);
    if (rsname[0] == '\0')
        strcpy(rsname, temp_ind);
    else
        strcpy(temp_ind, rsname);
    if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
    {
        choice = READI;
        chngd_x1_x2 = TRUE;
        chngd_y1_y2 = TRUE;
        chngd_w1_w2 = TRUE;
        chngd_w3_w4 = TRUE;
        read_ind(rs, up, rbai, sbai, index, nindex, &idumy4,
            &idumy1, &idumy2, &ddumy1, &idumy1,
            &ddumy2, &idumy3, &ddumy3, &all_ok, &idumy2);
        fclose(rs);
        if (all_ok)
        {
            if (sc == 0)
                g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
        }
        else
        {
            puts("\a");
            _outtext (mess8);
            gets(temp);
        }
    }
    else
    {
        puts("\a");
        gets(temp);
    }
    break;
case 'T' :
case 't' :
    _outtext(mess7);
    gets(rsname);
    if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
    {
        choice = READT;
        chngd_x1_x2 = TRUE;
        chngd_y1_y2 = TRUE;
        chngd_w1_w2 = TRUE;
        chngd_w3_w4 = TRUE;
        read_tek(rs, rbai, &idumy4, &idumy1, &idumy2,
            &ddumy1, &idumy1);
        fclose(rs);
    }

```

Source-code listing for program PLOT, continued.

```

        if (sc == 0)
            g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
        }
    else
    {
        puts("\a");
        gets(temp);
    }
    if (sc == 0)
        g_line(j,xadd, yadd, xscale, yscale, idumy4,
            rbai, vmode, 1);
    else
    {
        /* DO PLOT */
        plt_lne(rbai, idumy1, idumy4, x2, y2, 1.0, &port);
        cls();
    }
    break;
case '0' :
case 'o' :
    _outtext(mess9);
    gets(rsname);
    if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
    {
        choice = READD;
        chngd_x1_x2 = TRUE;
        chngd_y1_y2 = TRUE;
        chngd_w1_w2 = TRUE;
        chngd_w3_w4 = TRUE;
        read_oth(rs, rbai, &idumy4, &idumy1, &idumy2,
            &idumy1, &idumy1, &all_ok);
        fclose(rs);
        if (sc == 0)
            g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
    }
    else
    {
        puts("\a");
        gets(temp);
    }
    if (sc == 0)
        g_line(j,xadd, yadd, xscale, yscale, idumy4,
            rbai, vmode, 1);
    else
    {
        /* DO PLOT */
        plt_lne(rbai, idumy1, idumy4, x2, y2, 1.0, &port);
        cls();
    }
    gets(temp);
    break;
case 'p' :
case 'P' :
    plot_stp(x2, y2, &port);
    break;
default :
    break;
} /* end switch choice4 */
_outtext("\nFiles[(C)hron (I)nd (T)ek (O)ther] (P)rev menu ");
} /* end while */
}
break;
case 21 : /* ring-widths */
    Cdone = FALSE;
    while (!Cdone)
    {
        cls();
        _setvideomode(tmode);
        sc = menu(crow, ccol, mnuDevice);
    }

```

Source-code listing for program PLOT, continued.

```

if (sc == 3)
{
    Cdone = TRUE;
    break;
}
all_ok = TRUE;
if (choice == READ1)
{
    q4 = 0;
    for (i=1; i<=idumy4; i++)
    {
        if (up[i] > q4)
            q4 = up[i];
    }
    strcpy(y_lab, "WIDTH (mm)");
    Idone = FALSE;
    if (!chnegd_y1_y2)
    {
        q4 = 0;
        for (i=1; i<=idumy4; i++)
        {
            if (up[i] > q4)
                q4 = up[i];
        }
        y_tics(&t2, q4*10, &y2);
        y1 = y_min;
    }
    if (!chnegd_w3_w4)
    {
        w3 = y1;
        w4 = y2;
    }
    if (!chnegd_x1_x2)
        x_tics(&t1, &x1, &x2, idumy4, idumy1);
    if (!chnegd_w1_w2)
    {
        w1 = x1;
        w2 = x2;
    }
    if (sc == 0)
    {
        _setvideomode (vmode);
        yscale = (16.0*ys)/t2;
        xscale = (40.0*xs)/t1;
        g_labs(title, x_lab, y_lab, x1, y2, t1, t2, 2, 0);
        g_tics(xs, ys, 4.0, 20, 16.0, 7, w1, w2, w3, w4,
              xscale, yscale, x1, x2, y1, y2, t1, t2);
        g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
    }
    else
    {
        /* DO PLOT */
        _setvideomode (_DEFAULTMODE);
        p_plot(title, x_lab, y_lab, x1, x2, y1, y2, t1, t2, 4,
              FALSE, w1, w2, w3, w4, sc, &port, FALSE);
        _outtext (mess3);
    }
    while (!Idone)
    {
        if (all_ok)
        {
            if (sc == 0)
            {
                g_xyadd (&xadd, &yadd, idumy1, x1, y2, &j);
                g_line(j, xadd, yadd, xscale, yscale, idumy4,
                      "up, vmode, 10);
            }

```

Source-code listing for program PLOT, continued.

```

else
{
/* DO PLOT */
plt_lne(up, idumy1, idumy4, x2, y2, 10.0, &port);
cls();
}
}
_outtext("\nFiles[(1)nd] (P)rev menu\n");
c = getch();
switch (c)
{
case 'p' :
case 'P' :
plot_stp(x2, y2, &port);
ldone = TRUE;
break;
case 'l' :
case 'L' :
_outtext(mess6a);
_outtext(temp_ind);
_outtext(mess6b);
gets(rsname);
if (rsname[0] == '\0')
strcpy(rsname, temp_ind);
else
strcpy(temp_ind, rsname);
if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
{
choice = READI;
chngd_x1_x2 = TRUE;
chngd_y1_y2 = TRUE;
chngd_w1_w2 = TRUE;
chngd_w3_w4 = TRUE;
read_ind(rs, up, rbai, sbai, index, nindex, &idumy4,
&idumy1, &idumy2, &ddumy1, cdummy1,
&ddumy2, &idumy3, &ddumy3, &all_ok,
cdumy2);
fclose(rs);
if (all_ok)
{
}
else
{
puts("\a");
_outtext(mess8);
gets(temp);
}
} /* end good file */
else
{
puts("\a");
gets(temp);
}
break;
default :
break;
}
} /* end while */
} /* end if READI */
else
{
_setvideomode(DEFAULTMODE);
_outtext("\nOnly Individual (.IND) files contain ring-widths.\n");
gets(temp);
cdone = TRUE;
}
strcpy(y_lab, "BAI");
chngd_x1_x2 = FALSE;
chngd_y1_y2 = FALSE;

```

Source-code listing for program PLOT, continued.

```

        chngd_w1_w2 = FALSE;
        chngd_w3_w4 = FALSE;
    }
    break;
case 22 : /* indices */
    Cdone = FALSE;
    while (!Cdone)
    {
        cls();
        _setvideomode(tmode);
        indtyp = 0;
        indtyp = menu(crow, ccol, mnuindtyp);
        cls();
        if (indtyp == 2)
        {
            Cdone = TRUE;
            break;
        }
        sc = menu(crow, ccol, mnuDevice);
        cls();
        if (sc == 3)
        {
            Cdone = TRUE;
            break;
        }
        if (choice != READT)
        {
            if (choice == READD || choice == KYBD)
                for (i=0; i<=idumy4; i++)
                    index[i] = rbei[i];
            strcpy(y_lab, "INDEX");
            if (!chngd_w3_w4)
            {
                w3 = y1;
                if (indtyp == 0)
                    w4 = 2;
                else
                    w4 = 4;
            }
            if (!chngd_x1_x2)
                x_ticks(&t1, &x1, &x2, idumy4, idumy1);
            if (!chngd_w1_w2)
            {
                w1 = x1;
                w2 = x2;
            }
            if (sc == 0)
            {
                _setvideomode(vmode);
                if (indtyp == 0)
                    yscale = 56*ys;
                else
                    yscale = 18.66*ys;
                xscale = (40.0*xs)/t1;
                if (indtyp == 0)
                {
                    g_labs(title, x_lab, y_lab, x1, 2, t1, 1, 7, 0);
                    g_ticks(xs, ys, 5.6, 20, 55.75, 0, w1, w2, w3, w4,
                        xscale, yscale, x1, x2, y1, 2, t1, 1);
                }
                else
                {
                    g_labs(title, x_lab, y_lab, x1, 2, t1, 1, 7, 1);
                    g_ticks(xs, ys, 1.866, 60, 18.66, 6, w1, w2, w3, w4,
                        xscale, yscale, x1, x2, y1, 4, t1, 1);
                }
            }
            g_xyadd (&xadd, &yadd, idumy1, x1, 2, &j);
            _setlinestyle(style[12]);

```

Source-code listing for program PLOT, continued.

```

if(indtyp == 0)
    _moveto(0, (int)(55.5*ys));
    _lineto((int)(312*xs), (int)(55.5*ys));
}
else
{
    _moveto(0, (int)(37.32*ys));
    _lineto((int)(312*xs), (int)(37.32*ys));
    _moveto(0, (int)(74.64*ys));
    _lineto((int)(312*xs), (int)(74.64*ys));
}
setlinestyle(style[15]);
}
else
{
    /* DO PLOT */
    if(indtyp == 0)
        p_plot(title, x_lab, y_lab, x1, x2, y1, 2, t1, 1, 10,
            TRUE, w1, w2, w3, w4, sc, &port, indtyp);
    else
        p_plot(title, x_lab, y_lab, x1, x2, -3, 3, t1, 1, 10,
            TRUE, w1, w2, -3, 3, sc, &port, indtyp);
}
}
ldone = FALSE;
all_ok = TRUE;
while (!ldone)
{
    if (all_ok)
    {
        if(indtyp == 0)
            yadd = 2;
        else
            yadd = 4;
        if (sc == 0)
        {
            if (indtyp == 0)
            {
                g_xyadd (&xadd, &yadd, idumy1, x1, 2, &j);
                g_line(j, xadd, yadd, xscale, yscale, idumy4,
                    index, vmode, 1);
            }
            else
            {
                g_xyadd (&xadd, &yadd, idumy1, x1, 3, &j);
                g_line(j, xadd, yadd, xscale, yscale, idumy4,
                    nindex, vmode, 1);
            }
        }
        else
        {
            /* DO PLOT */
            if(indtyp == 0)
                plt_line(index, idumy1, idumy4, x2, 2,
                    1.0, &port);
            else
                plt_line(nindex, idumy1, idumy4, x2, 3,
                    1.0, &port);
            cls();
        }
    }
    _outtext("\nFiles[(C)hron (I)nd (O)ther] (P)rev menu\n");
    c = getch();
    if (c == 'p' || c == 'P')
    {
        plot_stp(x2, y2, &port);
        ldone = TRUE;
        break;
    }
}

```

Source-code listing for program PLOT, continued.

```

else
{
    if (c == 'c' || c == 'C')
    {
        _outtext(mess5);
        gets(rsname);
        all_ok = TRUE;
        if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
        {
            choice = READS;
            chngd_x1_x2 = TRUE;
            chngd_w1_w2 = TRUE;
            chngd_w3_w4 = TRUE;
            read_crn(rs, rings, rbai, sbai, up, low, index, nindex,
                &idumy4, &idumy1, &idumy2, &ddumy1,
                &idumy1, &all_ok, &ddumy2);
            fclose(rs);
            if (all_ok)
            {
            }
            else
            {
                puts("\a");
                _outtext(mess8);
                gets(temp);
            }
        }
        else
        {
            puts("\a");
            gets(temp);
        }
    }
    else
    {
        if (c == 'I' || c == 'i')
        {
            _outtext(mess6a);
            _outtext(temp_ind);
            _outtext(mess6b);
            gets(rsname);
            all_ok = TRUE;
            if (rsname[0] == '\0')
                strcpy(rsname, temp_ind);
            else
                strcpy(temp_ind, rsname);
            if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
            {
                choice = READI;
                chngd_x1_x2 = TRUE;
                chngd_w1_w2 = TRUE;
                chngd_w3_w4 = TRUE;
                read_ind(rs, up, rbai, sbai, index, nindex, &idumy4,
                    &idumy1, &idumy2, &ddumy1, &idumy1,
                    &ddumy2, &idumy3, &ddumy3, &all_ok,
                    &idumy2);
                fclose(rs);
                if (all_ok)
                {
                }
                else
                {
                    puts("\a");
                    _outtext(mess8);
                    gets(temp);
                }
            }
        }
    }
}

```

Source-code listing for program PLOT, continued.

```

else
(
puts("\a");
gets(temp);
)
/* end of if i */
else
(
if (c == '0' || c == 'o')
(
_outtext(mess9);
gets(rsname);
all_ok = TRUE;
if ((rs = setupinf(rsname, "r")) != (FILE *)NULL)
(
choice = READT;
read_oth(rs, index, &idumy4, &idumy1, &idumy2,
&ddumy1, &cdumy1, &all_ok);
fclose(rs);
if (all_ok)
(
)
else
(
puts("\a");
_outtext (mess8);
gets(temp);
)
)
else
(
puts("\a");
gets(temp);
)
)
)
/* end check i */
/* end check input */
/* end idone */
)
else
(
puts("\a");
_outtext ("\nCannot be TEKTRONICS file.\n");
gets(temp);
)
strcpy(y_lab, "9A1");
)
chngd_x1_x2 = FALSE;
chngd_y1_y2 = FALSE;
chngd_w1_w2 = FALSE;
chngd_w3_w4 = FALSE;
break;
)
/* end ddone */
break;
case 3 : /* Listing routines */
lmdone = FALSE;
while (!lmdone)
(
cls();
lmenu = menu(crow, ccol, menu1main);
lmenu = lmenu+30;
switch(lmenu)
(
case 33 :
lmdone = TRUE;
break;

```

Source-code listing for program PLOT, continued.

```

case 30 : /* LDATAS */
  Ldone = FALSE;
  while (!Ldone)
  {
    cls();
    lpt = menu(crow, ccol, mnuldata);
    if (lpt == 2)
    {
      Ldone = TRUE;
      break;
    }
    cls();
    if (lpt == 0)
    {
      if (choice == READS)
        list_crn(first_yr, years, rings, rbai, sbai, up,
          low, index, nindex, title);
      else
      {
        if (choice == READI)
          list_ind(first_yr, years, up, rbai, sbai, index,
            nindex, title, core);
        else
        {
          if (choice > READI && choice < DIRL)
            list_tek(rbai, years, first_yr, title);
        }
      }
    }
    else
    {
      if (choice == READS)
        prnt_crn(first_yr, years, rings, rbai, sbai,
          up, low, index, nindex, title);
      else
      {
        if (choice == READI)
          prnt_ind(first_yr, years, up, rbai, sbai, index,
            nindex, title);
        else
        {
          if (choice > READI && choice < DIRL)
            prnt_tek(rbai, years, first_yr, title);
        }
      }
    }
  }
  break;
case 31 : /* LSTAT */
  Ldone = FALSE;
  while (!Ldone)
  {
    cls();
    lpt = menu(crow, ccol, mnuldata);
    if (lpt == 2)
    {
      Ldone = TRUE;
      break;
    }
    if (lpt == 0)
      list_inf(title, x_lab, y_lab, first_yr, last_yr, y_max,
        q1, q2, q3, x1, x2, y2, w1, w2, w3, w4, ms,
        cyr, cval, choice);
    else
      prnt_inf(title, x_lab, y_lab, first_yr, last_yr, y_max,
        q1, q2, q3, x1, x2, y2, w1, w2, w3, w4, ms,
        cyr, cval, choice);
  }

```

Source-code listing For program PLOT, continued.

```

        gets(temp);
    }
    break;
case 32 :
    ldone = FALSE;
    while (!ldone)
    {
        cls();
        lpt = menu(crow, ccol, mnuldata);
        if (lpt == 2)
        {
            ldone = TRUE;
            break;
        }
        if (lpt < 2)
        {
            cls();
            dirnam[0] = '\0';
            printf("\n\nEnter file name (including full path): ");
            gets(temp_dir);
            if (lpt == 0)
            {
                strcpy(dirnam, "type ");
                strcat(dirnam, temp_dir);
                strcat(dirnam, " | more");
            }
            else
            {
                strcpy(dirnam, "print ");
                strcat(dirnam, temp_dir);
            }
            cls();
            i = system(dirnam);
            cntrue();
            gets(temp_dir);
        }
    }
    break;
} /* end switch (menu) */

} /* end while ldone */
break;
case 4 : /* SKEYF */
if (choice == 4)
{
    all_ok = TRUE;
    _outtext(mess9);
    gets(rsname);
    if ((rs = setupout(rsname, "w")) != (FILE *)NULL)
    {
        fprintf(rs, "%s\n\n", title);
        writef(rs, rbal, years*1, first_yr, last_yr);
        fclose(rs);
        if (all_ok)
        {
            puts("\a");
            _clearscreen(0);
            _outtext (mess10);
            _outtext (mess3);
            ldone = TRUE;
        }
    }
    else
    {
        puts("\a");
        _outtext (mess8);
        _outtext (" File may be wrong type.");
        _outtext (mess3);
    }
}
}

```

Source-code listing for program PLOT, continued.

```

        else
        {
            puts("\a");
            _outtext (mess3);
        }
    }
    else
    {
        _outtext("\nOnly Keyboard entries can be saved.");
    }
    gets(temp);
    break;
case 5 : /* MAIN */
    Pdone = TRUE;
    break;
default :
    break;
}
_setvideomode (tmode);
}
/* end pmenu */
} /* end main for */
}

/* Put menu on screen.
   Starting <row> and <column>.
   Array of menu <items> strings.
   Global structure variable <menus> determines:
       Colors of border, normal items, and selected item.
       Centered or left justified.
       Border characters.
   Returns number of item selected. */

int menu(row, col, items)
int row, col;
char *items[];
{
    int i, num, max = 2, prev, curr = 0, choice, c, d;
    int litem[25];
    long bcolor;

    cursor(TOUCRSOROFF);
    bcolor = _getbkcolor();

    /* Count items, find longest, and put length of each in array */
    for (num = 0; items[num]; num++) {
        litem[num] = strlen(items[num]);
        max = (litem[num] > max) ? litem[num] : max;
    }
    max += 2;

    if (menus.centered) {
        row -= num / 2;
        col -= max / 2;
    }

    /* Draw      menu box */

    _settextcolor(menus.fgBorder);
    _setbkcolor(menus.bgBorder);
    box(row++, col++, num, max);

    /* Put items in menu */

    for (i = 0; i < num; ++i) {
        if (i == curr) {
            _settextcolor(menus.fgSelect);
            _setbkcolor(menus.bgSelect);

```

Source-code listing *for program PLOT, continued.

```

    } else {
        _settextcolor(menus.fgNormal);
        _setbkcolor(menus.bgNormal);
    }
    itemize(row+i,col,items[i],max - litem[i]);
}

/* Get selection */
for (;;) {
    switch (c = ((_bios_keybrd(_KEYBRD_READ) & 0xff00) >> 8))
    {
        case UP:
            prev = curr;
            curr = (curr > 0) ? (--curr % num) : num-1;
            break;
        case DOWN:
            prev = curr;
            curr = (curr < num) ? (++curr % num) : 0;
            break;
        case ENTER:
            _setbkcolor(bcolor);
            return(curr);
        default:
            if (c > 0 && c < 12)
            {
                c = c-2;
                if (c < num)
                    return (c);
            }
            continue;
    }
    _settextcolor(menus.fgSelect);
    _setbkcolor(menus.bgSelect);
    itemize(row+curr,col,items[curr],max - litem[curr]);
    _settextcolor(menus.fgNormal);
    _setbkcolor(menus.bgNormal);
    itemize(row+prev,col,items[prev],max - litem[prev]);
}

/* Draw menu box.
   <row> and <col> are upper left of box.
   <hi> and <wid> are height and width. */
void box(row, col, hi, wid)
int row, col, hi, wid;
{
    int i;
    char temp[80];

    _settextposition(row,col);
    temp[0] = *menus.nw;
    memset(temp+1,*menus.ew,wid);
    temp[wid+1] = *menus.ne;
    temp[wid+2] = NULL;
    _outtext(temp);
    for (i = 1; i <= hi; ++i) {
        _settextposition(row+i,col);
        _outtext(menus.ns);
        _settextposition(row+i,col+wid+1);
        _outtext(menus.ns);
    }
    _settextposition(row+hi+1,col);
    temp[0] = *menus.sw;
    memset(temp+1,*menus.ew,wid);
    temp[wid+1] = *menus.se;
    temp[wid+2] = NULL;

```

Source-code listing for program PLOT, continued.

```

        _outtext(temp);
    }

    /* Put an item in menu.
       <row> and <col> are left position.
       <str> is the string item.
       <len> is the number of blanks to fill. */

void itemize(row,col,str,len)
int row, col, len;
char str[];
{
    char temp[80];

    _settextposition(row,col);
    _outtext(" ");
    _outtext(str);
    memset(temp, ' ', len--);
    temp[len] = NULL;
    _outtext(temp);
}

/* Change the cursor shape.
   <value> has starting line in upper byte, ending line in lower byte.
   Returns the previous cursor value. */

unsigned cursor(value)
unsigned value;
{
    union REGS inregs, outregs;
    int ret;

    inregs.h.ah = 3;          /* Get old cursor */
    inregs.h.bh = 0;
    int86(0x10,&inregs,&outregs);
    ret = outregs.x.cx;

    inregs.h.ah = 1;          /* Set new cursor */
    inregs.x.cx = value;
    int86(0x10,&inregs,&outregs);

    return(ret);
}

/*****
/* File name      : CHNG_ALL.C
*****/

#include <stdio.h>

#define TRUE 1
#define FALSE 0

chgng_t(title)

char title[];
{
    int done = FALSE;
    char temp[81];

    cls();
    printf("This is the current title:\n %s\n",title);
    while (!done)
    {
        printf("\nEnter the new title: \n");
        gets(temp);
        if (temp[0] != '\0')
            strcpy(title, temp);
    }
}

```

Source-code listing for program PLOT, continued.

```
    printf("This is the new Title: \n %s\n",title);
    printf("\n\n Is this correct (Y/N)? ");
    gets(temp);
    if (temp[0] == 'Y' || temp[0] == 'y' || temp[0] == '\0')
        done = TRUE;
    }
}

chg_x(x_lab)

char x_lab[];
{
    int done = FALSE;
    char temp[41];

    cls();
    puts("\n\nThis is the current X label: ");
    puts(x_lab);
    while (!done)
    {
        puts("\nEnter new X label: ");
        gets(temp);
        if (temp[0] != '\0')
            strcpy(x_lab, temp);
        puts("This is the new X label: ");
        puts(x_lab);
        puts("\n\n Is this correct (Y/N)? ");
        gets(temp);
        if (temp[0] == 'Y' || temp[0] == 'y');
            done = TRUE;
    }
}

chg_y(y_lab)

char y_lab[];
{
    int done = FALSE;
    char temp[41];

    cls();
    puts("\n\nThis is the current Y label: ");
    puts(y_lab);
    while (!done)
    {
        puts("\nEnter new Y label: ");
        gets(temp);
        if (temp[0] != '\0')
            strcpy(y_lab, temp);
        puts("This is the new Y label: ");
        puts(y_lab);
        puts("\n\n Is this correct (Y/N)? ");
        gets(temp);
        if (temp[0] == 'Y' || temp[0] == 'y');
            done = TRUE;
    }
}

chg_x_vals(x1, x2, t1)

int *x1, *x2, *t1;

{
    int done = FALSE;
    char temp[5];
    char tempx[5];
    char tempt[7];

    cls();
```

Source-code listing for program PLOT, continued.

```

puts("Current X axis info:\n");
printf("First year: %d    Last year: %d    Years/Major interval: %3d\n\n",
      *x1, *x2, *t1);
puts("    You will only be allowed to specify the first year");
puts("    and the years per Major interval.\n");
while (!done)
{
    printf("First year:  ");
    gets(tempx);
    printf("Interval(yrs): ");
    gets(temp);
    if (tempx[0] != '\0')
        *x1 = atoi(tempx);
    if (temp[0] != '\0')
        *t1 = atoi(temp);
    *x2 = *x1 + (*t1 * 7);
    puts("\nThe new values are:\n");
    printf("First year: %d    Last year: %d    Years/Major interval: %3d\n\n",
          *x1, *x2, *t1);
    puts("\n Is this correct (Y/N)? ");
    gets(temp);
    if (temp[0] != 'N' && temp[0] != 'n')
        done = TRUE;
}
}

chg_x_win(w1, w2, x1, x2)

int *w1, *w2, *x1, *x2;

{
    int done = FALSE;
    char temp[5];
    char tempx[5];
    char tempt[7];

    cls();
    puts("Current X window info:\n");
    printf("First year: %d    Last year: %d\n\n",
          *w1, *w2);
    while (!done)
    {
        printf("First year: ");
        gets(tempx);
        printf("Last year : ");
        gets(temp);
        if (tempx[0] != '\0')
            *w1 = atoi(tempx);
        if (*w1 < *x1)
            *w1 = *x1;
        if (temp[0] != '\0')
            *w2 = atoi(temp);
        if (*w2 > *x2)
            *w2 = *x2;
        puts("\nThe new values are:\n");
        printf("First year: %d    Last year: %d\n\n",
              *w1, *w2);
        printf("\n Is this correct (Y/N)? ");
        gets(temp);
        if (temp[0] != 'N' && temp[0] != 'n')
            done = TRUE;
    }
}

chg_y_vals(y1, y2, t2)

int *y1, *y2, *t2;

{

```

Source-code listing for program PLOT, continued.

```

int done = FALSE;
char temp[5];
char tempx[5];
char tempt[7];

cls();
puts("Current Y axis info:\n");
printf("Y minimum: %d    Y maximum: %d    Major interval: %d\n\n",
        *y1, *y2, *t2);
puts("    You will only be allowed to specify the Y maximum value");
puts("    and the value per Major interval.\n");
while (!done)
{
    printf("Y minimum: ");
    gets(tempx);
    printf("Interval : ");
    gets(tempt);
    if (tempx[0] != '\0')
        *y1 = atoi(tempx);
    if (tempt[0] != '\0')
        *t2 = atoi(tempt);
    *y2 = *t2 * 7;
    puts("\nThe new values are:\n");
    printf("Y minimum: %d    Y maximum: %d    Major interval: %d\n\n",
            *y1, *y2, *t2);
    printf("\n Is this correct (Y/N)? ");
    gets(temp);
    if (temp[0] != 'N' && temp[0] != 'n')
        done = TRUE;
}
}

chg_y_win(w3, w4, y1, y2)

int *w3, *w4, *y1, *y2;

{
    int done = FALSE;
    char temp[5];
    char tempx[5];
    char tempt[7];

    cls();
    puts("Current Y window info:\n");
    printf("Y minimum: %d    Y maximum: %d\n\n",
            *w3, *w4);
    while (!done)
    {
        printf("Y minimum: ");
        gets(tempx);
        printf("Y maximum: ");
        gets(tempt);
        if (tempx[0] != '\0')
            *w3 = atoi(tempx);
        if (*w3 < *y1)
            *w3 = *y1;
        if (tempt[0] != '\0')
            *w4 = atoi(tempt);
        if (*w4 > *y2)
            *w4 = *y2;
        puts("\nThe new values are:\n");
        printf("Y minimum: %d    Y maximum: %d\n\n",
                *w3, *w4);
        printf("\n Is this correct (Y/N)? ");
        gets(temp);
        if (temp[0] != 'N' && temp[0] != 'n')
            done = TRUE;
    }
}

```

Source-code listing for program PLOT, continued.

```

/*****
/* File name      : FIL_ARRS.C
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

fill_sarray(id, a, q2, first_yr, last_yr, years)

FILE *id;
double *a, *q2;
int *first_yr, *last_yr, *years;

{
char *cf, *fgets();
double atof();
int done, first, line_len, cnt, i, j;
char temp_str[10], line[80];

done = FALSE;
cnt = 1;
first = TRUE;
while (!done)
{
cf = fgets(line, 80, id);
line_len = strlen(line);
if ((line_len > 4) && (!feof(id)))
{
if (first)
{
substr(temp_str, line, 1, 4);
*first_yr = atoi(temp_str);
first = FALSE;
}
j = (line_len-4)/7;
for (i=1; i<=j; i++)
{
substr(temp_str, line, 5+((i-1)*7), 7);
a[cnt] = atof(temp_str);
if (a[cnt] > *q2)
*q2 = a[cnt];
cnt = cnt + 1;
}
}
else
{
done = TRUE;
*years = cnt-1;
*last_yr = *first_yr+*years-1;
}
}
}

fill_tarray(id, a, q2, first_yr)

FILE *id;
double *a, *q2;
int *first_yr;

{
char *cf, *fgets();
double atof();
int done, cnt, i, j, tyr;
char temp_str[10], line[80];
done = FALSE;
cnt = 1;
while (!done)
{
cf = fgets(line, 80, id);

```

Source-code listing for program PLOT, continued.

```

    substr(temp_str, line, 2, 4);
    tyr = atoi(temp_str);
    if ((tyr-*first_yr >= 0) && (!feof(id)))
    {
        substr(temp_str, line, 19, 6);
        a[cnt] = atof(temp_str);
        if (a[cnt] > *q2)
            *q2 = a[cnt];
        cnt = cnt + 1;
    }
    else
        done = TRUE;
}

/*****
/* File name      : LIST_ALL.C
*****/

#include <stdio.h>

#define TRUE 1
#define FALSE 0

list_crn(fyr, years, r, a, m, u, l, ia, in, title)
int fyr, years;
double *r, *a, *m, *u, *l, *ia, *in;
char title[];
{
    int i, line_cnt, rint;
    char temp[5];

    line_cnt = 4;
    printf("%s\n\n", title);
    printf("YEAR      AREA      MEAN      UPPER      LOWER      INDEX      WINDEX      CORES\n");
    for (i=1; i<=years; i++)
    {
        if (line_cnt > 23)
        {
            printf("
            (Press enter to continue.)");
            gets(temp);
            cls();
            printf("%s\n\n", title);
            printf("YEAR      AREA      MEAN      UPPER      LOWER      INDEX      WINDEX      CORES\n");
            line_cnt = 4;
        }
        rint = r[i];
        printf("%4d      %7.3f %7.3f %7.3f %7.3f %7.3f %7.3f %7d\n", fyr, a[i],
            m[i], u[i], l[i], ia[i], in[i], rint);
        line_cnt = line_cnt+1;
        fyr = fyr+1;
    }
    cntnue();
    gets(temp);
}

list_ind(fyr, years, a, m, u, l, in, title, core)
int fyr, years;
double *a, *m, *u, *l, *in;
char title[], core[];
{
    int i, line_cnt;
    char temp[5];

```

Source-code listing for program PLOT, continued.

```

line_cnt = 4;
printf("%s ( %s )\n\n", title, core);
printf("YEAR    WIDTH    AREA    SPLINE    INDEX    NINDEX\n");
for (i=1; i<=years; i++)
{
    if (line_cnt > 23)
    {
        printf("                                (Press enter to continue.)");
        gets(temp);
        cls();
        printf("%s ( %s )\n\n", title, core);
        printf("YEAR    WIDTH    AREA    SPLINE    INDEX    NINDEX\n");
        line_cnt = 4;
    }
    printf("%4d    %7.3f %7.3f %7.3f %7.3f %7.3f\n", fyr, a[i],
        m[i], u[i], l[i], in[i]);
    line_cnt = line_cnt+1;
    fyr = fyr+1;
}
cntnue();
gets(temp);
}

list_tek(prtarr, j, fyr1, title)
int j, fyr1;
double prtarr[];
char title[];

{
    int l, i, jb, je, line_cnt;
    int done = FALSE;
    char temp[5];
    jb = 1;
    je = (10 - (fyr1 % 10));
    line_cnt = 0;
    cls();
    printf("%s\n\n", title);
    while (!done)
    {
        if (i == j+1)
        {
            done = TRUE;
            break;
        }
        printf("%4d", fyr1);
        for (i=jb; i <= je; i++)
        {
            if (i == j+1)
            {
                done = TRUE;
                break;
            }
            printf("%7.3f", prtarr[i]);
        }
        printf("\n");
        if (i <= 10)
            fyr1 = fyr1 + i - 1;
        else
            fyr1 = fyr1 + 10;
        jb = je + 1;
        je = je + 10;
        line_cnt = line_cnt+1;
        if (line_cnt > 23)
        {
            line_cnt = 1;
            cntnue();
            gets(temp);
            cls();
            printf("\n%s\n\n", title);
        }
    }
}

```

Source-code listing for program PLOT, continued.

```

    }
    cntnue();
    gets(temp);
}
list_inf(title, x_lab, y_lab, fyr, lyr, y_max, q1, q2, q3,
        x1, x2, y2, w1, w2, w3, w4, ms, cyr, cval, choice)

char *title[], *x_lab[], *y_lab[];
int fyr, lyr, q1, q2, q3, x1, x2, y2, w1, w2, w3, w4, cyr, choice;
double y_max, ms, cval;

{
    char temp[5];

    cls();
    printf("\n\n\n          %s\n", title);
    printf("      X Label: %s\n", x_lab);
    printf("      Y Label: %s\n", y_lab);
    printf("\n          DATA VALUES\n");
    printf("      First Year   : %4d\n", fyr);
    printf("      Last Year    : %4d\n", lyr);
    printf("      Years        : %4d\n", (lyr-fyr)+1);
    printf("      Maximum Y     : %7.3f\n", y_max);
    if (choice == 0 || choice == 1)
        printf("      M.S.         : %6.3f\n", ms);
    if (choice == 1)
        printf("                  %4d = %7.3f Cm.\n", cyr, cval);
    else
        printf("\n");
    printf("\n          PLOT VALUES\n");
    printf("          DEFAULT   CURRENT\n");
    printf("          AXIS/WIND  AXIS      DATA WINDOW\n");
    printf("      Min X axis    : %4d      %4d      %4d\n", q1, x1, w1);
    printf("      Max X axis    : %4d      %4d      %4d\n", q2, x2, w2);
    printf("      X units       : %4d      %4d\n", (q2-q1)/7, (x2-x1)/7);
    printf("      Min Y axis    : 0        0        %4d\n", w3);
    printf("      Max Y axis    : %4d      %4d      %4d\n", q3, y2, w4);
    printf("      Y units       : %4d      %4d\n", q3/7, y2/7);
    cntnue();
}

/*****
/* File name      . PLT_UTL.C
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

cls()
{
    printf("\033[2J");
    fflush(stdout);
}

cntnue()
{
    char temp[2];

    printf("          (Press enter to continue.)");
}

index(s, t)
char s[], t[];
{
    int i, j, k;

```

Source-code listing for program PLOT, continued.

```

for(i=0; s[i] != '\0'; i++)
{
    for(j=i, k=0; t[k] != '\0' && s[j]==t[k]; j++, k++);
    if (t[k] == '\0') return(i);
}
return(-1);
}

y_tics(t2, q2, y2)
int *t2, *y2;
double q2;

{
    *t2 = (q2/7)+1;
    *y2 = *t2 * 7;
}

x_tics(t1, x1, x2, years, fyr)
int *t1, *x1, *x2;
int years, fyr;

{
    int diff, fdec;
    double t1_temp;

    t1_temp = (double)(years)/30.0;
    if (t1_temp < 1.0)
    {
        *t1 = 5;
    }
    else
    {
        *t1 = ((int)(t1_temp)+1)*5;
    }
    diff = (*t1*7) - years;
    *x1 = (fyr-(diff/2));
    *x2 = *x1+(*t1*7);
}

substr(substring, string, start, length)
char *substring;
char *string;
int start;
int length;
{
    int i;
    start--;
    string += start;
    for (i=0; i < length && *string != '\n' && *string != '\0'; i++)
        *substring++ = *string++;
    *substring = '\0';
}

/*****
/* File name      : PRINT_ALL.C          */
*****/

#include <stdio.h>

#define TRUE 1
#define FALSE 0

prnt_crn(fyr, years, r, a, m, u, l, ia, in, title)

```

Source-code listing for program PLOT, continued.

```

int fyr, years;
double *r, *a, *m, *u, *l, *ia, *in;
char title[];

(
int i, line_cnt, rint;
char temp[5];

line_cnt = 6;
fprintf(stdprn, "\n\n          %s\n\n", title);
fprintf(stdprn, "          YEAR      AREA      MEAN      UPPER      LOWER      INDEX      WINDEX      CORES\n");
for (i=1; i<=years; i++)
(
if (line_cnt > 56)
(
fprintf(stdprn, "\f\n\n          %s\n\n", title);
fprintf(stdprn, "          YEAR      AREA      MEAN      UPPER      LOWER      INDEX      WINDEX      CORES\n");
line_cnt = 6;
)
rint = r[i];
fprintf(stdprn, "          %4d      %7.3f %7.3f %7.3f %7.3f %7.3f %7.3f %7.3f %7d\n",
fyr, a[i], m[i], u[i], l[i], ia[i], in[i], rint);
line_cnt = line_cnt+1;
fyr = fyr+1;
)
fprintf(stdprn, "\f");
continue();
)

prnt_ind(fyr, years, a, m, u, l, in, title, core)

int fyr, years;
double *a, *m, *u, *l, *in;
char title[], core[];

(
int i, line_cnt;
char temp[5];

line_cnt = 6;
fprintf(stdprn, "\n\n          %s ( %s )\n\n", title, core);
fprintf(stdprn, "          YEAR      WIDTH      AREA      SPLINE      INDEX      WINDEX\n");
for (i=1; i<=years; i++)
(
if (line_cnt > 56)
(
fprintf(stdprn, "\f\n\n          %s ( %s )\n\n", title, core);
fprintf(stdprn, "          YEAR      WIDTH      AREA      SPLINE      INDEX      WINDEX\n");
line_cnt = 6;
)
fprintf(stdprn, "          %4d      %7.3f %7.3f %7.3f %7.3f %7.3f\n", fyr, a[i],
m[i], u[i], l[i], in[i]);
line_cnt = line_cnt+1;
fyr = fyr+1;
)
fprintf(stdprn, "\f");
continue();
)

prnt_tek(prtarr, j, fyr1, title)
int j, fyr1;
double prtarr[];
char title[];

(
int l, i, jb, je, line_cnt;
int done = FALSE;

```

Source-code listing for program PLOT, continued.

```
char temp[5];
jb = 1;
je = (10 - (fyr1 % 10));
line_cnt = 6;
fprintf(stdprn, "\n\n   %s\n\n", title);
while (!done)
{
    if (i == j+1)
    {
        done = TRUE;
        break;
    }
    fprintf(stdprn, "      %4d", fyr1);
    for (i=jb; i <= je; i++)
    {
        if (i == j+1)
        {
            done = TRUE;
            break;
        }
        fprintf(stdprn, "%7.3f", prtarr[i]);
    }
    fprintf(stdprn, "\n");
    if (i <= 10)
        fyr1 = fyr1 + i - 1;
    else
        fyr1 = fyr1 + 10;
    jb = je + 1;
    je = je + 10;
    line_cnt = line_cnt+1;
    if ((line_cnt > 56))
    {
        line_cnt = 6;
        fprintf(stdprn, "\f\n\n   %s\n\n", title);
    }
}
fprintf(stdprn, "\f");
cnttrue();
)

print_inf(title, x_lab, y_lab, fyr, lyr, y_max, q1, q2, q3,
          x1, x2, y2, w1, w2, w3, w4, ms, cyr, cval, choice)

char *title[], *x_lab[], *y_lab[];
int fyr, lyr, q1, q2, q3, x1, x2, y2, w1, w2, w3, w4, cyr, choice;
double y_max, ms, cval;

(

char temp[5];

cls();
fprintf(stdprn, "\n\n\n           %s\n\n", title);
fprintf(stdprn, "X Label: %s\n", x_lab);
fprintf(stdprn, "Y Label: %s\n\n", y_lab);
fprintf(stdprn, "\n                                DATA VALUES\n");
fprintf(stdprn, "First Year       : %4d", fyr);
fprintf(stdprn, "Last Year        : %4d\n", lyr);
fprintf(stdprn, "Years           : %4d", (lyr-fyr)+1);
fprintf(stdprn, "Maximum Y       :%7.3f\n", y_max);
if (choice == 0 || choice == 1)
    fprintf(stdprn, "W.S.             :%7.3f", ms);
if (choice == 1)
    fprintf(stdprn, "                  %4d =%7.3f Om.\n", cyr, cval);
else
    fprintf(stdprn, "\n");
fprintf(stdprn, "\n                                PLOT VALUES\n");
fprintf(stdprn, "                                DEFAULT CURRENT\n");
fprintf(stdprn, "                                AXIS WINDOW\n");
```

Source-code listing for program PLOT, continued.

```

printf(stdprn, "Min X axis : %d\n", q1, x1, w1);
printf(stdprn, "Max X axis : %d\n", q2, x2, w2);
printf(stdprn, "X units : %d\n", (q2-q1)/7, (x2-x1)/7);
printf(stdprn, "Min Y axis : 0\n", 0, w3);
printf(stdprn, "Max Y axis : %d\n", q3, y2, w4);
printf(stdprn, "Y units : %d\n", q3/7, y2/7);
continue();
}

/* File name : P_PLOT.C */

#include <conio.h>
#include <string.h>
#include <stdio.h>
#include <dos.h>
#include <bios.h>

#define XOM 0x11
#define XOFF 0x13
#define TRUE 1
#define FALSE 0

p_plot(title, x_lab, y_lab, x1, x2, y1, y2, t1, t2, t3, idx, w1, w2, w3, w4,
choice, port, indtyp)

char title[], x_lab[], y_lab[];
int x1, x2, y1, y2, t1, t2, t3, idx, choice;
int w1, w2, w3, w4, indtyp;
int *port;
{
    unsigned incr;
    char plot_string[500], y_lab[17];
    int i, j, p1, p2, p3, p4;
    double x3, x4, y3, y4;

    strcpy(y_lab, "NORMALIZED INDEX");
    _bios_serialcom(_COM_INIT, 0, (_COM_CHR8 | _COM_STOP1 | _COM_EVENPARITY |
    _COM_9600));
    plot_str("\x1B.181;17:\x1B.N;19:", port);
    if (choice == 1)
    {
        sprintf(plot_string, "IN;PS4;IP1200,1200,9500,6750;SC%d,%d,%d,%d",
            x1, x2, y1, y2);
        plot_str(plot_string, port);
        p1 = (1200+((8300/(x2-x1))*(w1-x1)));
        p2 = (9500-((8300/(x2-x1))*(x2-w2)));
        p3 = (1200+((5550/(y2-y1))*(w3-y1)));
        p4 = (6750-((5550/(y2-y1))*(y2-w4)));
    }
    else
    {
        sprintf(plot_string, "IN;PS0;IP2560,1260,14300,6750;SC%d,%d,%d,%d",
            x1, x2, y1, y2);
        plot_str(plot_string, port);
        p1 = (2560+((11740/(x2-x1))*(w1-x1)));
        p2 = (14300-((11740/(x2-x1))*(x2-w2)));
        p3 = (1260+((7890/(y2-y1))*(w3-y1)));
        p4 = (6750-((7890/(y2-y1))*(y2-w4)));
    }
    sprintf(plot_string, "PU;SP1;PA %d,%d;SI .3,.5;CP%d,1.0;LB%3",
        (x2+x1)/2, y2, -(strlen(title)/2), title);
    plot_str(plot_string, port);
    if (indtyp == 0)
        sprintf(plot_string, "PU;PA%d,%d;SI.3,.5;D10,1;CP%d,2.0;LB%3",
            x1, (y2-y1)/2, -(strlen(y_lab)/2)+1, y_lab);
}

```

Source-code listing for program PLOT, continued.

```

else
    sprintf(plot_string, "PU;PA%d,0;SI.3,.5;D1C,1;CP%d,2.0;LB%2s\3",
        x1, -(strlen(y_lab1)/2)+1, y_lab1);
    plot_str(plot_string, port);
    sprintf(plot_string, "PU;D11,0;PA%d,%d;SI.3,.5;CP%d,-1.5;LB%2s\3",
        (x2+x1)/2, y1, -(strlen(x_lab)/2), x_lab);
    plot_str(plot_string, port);
    sprintf(plot_string, "PU;SP2;PA%d,%d;PD;PA%d,%d,%d,%d,%d,%d,%d",
        x1, y1, x1, y2, x2, y2, x2, y1, x1, y1);
    plot_str(plot_string, port);
    sprintf(plot_string, "PU;SI.2,.3;TL1.5,0");
    plot_str(plot_string, port);

/* LOWER X TICS AND AXIS LABEL */
x3 = x1;
x4 = (double)(t1)/5;
for (i=x1; i<=x2; i+=t1)
{
    sprintf(plot_string, "PU;TL0,1.5;PA%d,%d;CP-2.0,-1.0;LB%4d\3", i, y1, i);
    plot_str(plot_string, port);
    if (i<x2 && i>x1)
    {
        sprintf(plot_string, "PU;TL1.5,0;PA%d,%d;XT", i, y1);
        plot_str(plot_string, port);
    }
    for (j=1; j<=5; j++)
    {
        if (x3<=i+t1 && x3>i && x3<x2)
        {
            sprintf(plot_string, "PU;TL.75,0;PA%6.2f,%d;XT", x3, y1);
            plot_str(plot_string, port);
        }
        x3 = x3+x4;
    }
}

/* RIGHT Y TICS */
y3 = y1;
if (idx && indtyp == 0)
    y4 = 0.1;
else
    y4 = (double)(t2)/(double)(t3);
for (i=y1; i<=y2; i+=t2)
{
    if ((i>y1 && i<y2) && !idx)
    {
        sprintf(plot_string, "PU;TL0,1.25;PA%d,%d;YT", x2, i);
        plot_str(plot_string, port);
    }
    for (j=1; j<=t3; j++)
    {
        if ((y3>i && y3<i+t2) && y3<y2)
        {
            sprintf(plot_string, "PU;TL0,.5;PA%d,%6.2f;YT", x2, y3);
            plot_str(plot_string, port);
        }
        y3 = y3+y4;
    }
}

/* LEFT TICS AND LABELS */
y3 = y1;
for (i=y1; i<=y2; i+=t2)
{
    if (i > 9)
        sprintf(plot_string, "PU;TL1.5,0;PA%d,%d;CP-3.0,-.2;LB%2d\3",
            x1, i, i);
}

```

Source-code listing for program PLOT, continued.

```

else
    sprintf(plot_string, "PU;TL1.5,0;PA%d,%d;CP-2.0,-.2;LB%1d\3",
        x1, i, i);
    plot_str(plot_string, port);
    if ((i>y1 && i<y2) && !idx)
    {
        sprintf(plot_string, "PU;TL1.25,0;PA%d,%d;YT", x1, i);
        plot_str(plot_string, port);
    }
    for (j=1; j<=t3; j++)
    {
        if ((y3>i && y3<i+t2) && y3<y2)
        {
            sprintf(plot_string, "PU;TL.5,0;PA%d,%6.2f;YT", x1, y3);
            plot_str(plot_string, port);
        }
        y3 = y3+y4;
    }
}

/* UPPER X TICS */
x3 = x1;
for (i=x1; i<=x2; i+=t1)
{
    if (i<x2 && i>x1)
    {
        sprintf(plot_string, "PU;TL0,1.5;PA%d,%d;XT", i, y2);
        plot_str(plot_string, port);
    }
    for (j=0; j<=4; j++)
    {
        if (x3<i+t1 && x3>i && x3<x2)
        {
            sprintf(plot_string, "PU;TL0,.75;PA%6.2f,%d;XT", x3, y2);
            plot_str(plot_string, port);
        }
        x3 = x3+x4;
    }
}
if (idx)
{
    sprintf(plot_string, "PU;LT3,4;SP3;PA%d,1.0;PD;PA%d,1.0;PU;LT;SP0", x1, x2);
    plot_str(plot_string, port);
}
if (indtyp == 1)
{
    sprintf(plot_string, "PU;LT3,4;SP3;PA%d,-1.0;PD;PA%d,-1.0;PU;LT;SP0", x1, x2);
    plot_str(plot_string, port);
}
sprintf(plot_string, "IW%d,%d,%d,%d;PU;SP0;PA%d,%d", p1, p3, p2, p4, x2, y2);
plot_str(plot_string, port);
int *port;
char *plot_string;
{
    unsigned far *com_base;
    char outchar;
    int out_element;

    if (*port == 0)
        com_base = (unsigned far *) 0x00000400L;
    else
        com_base = (unsigned far *) 0x00000402L;
    for (out_element = 0; out_element < strlen(plot_string); ++out_element)
    {
        if ((unsigned char) inp(*com_base + 5) & 0x01)
            if ((unsigned char) inp(*com_base) == 0xFF)

```

Source-code listing for program PLOT, continued.

```

        while (((unsigned char)inp(*com_base) != XON);
        while (((unsigned char)inp(*com_base + 5) & 0x20) == NULL);
        outp(*com_base, plot_string[out_element]);
    }
    strcpy(plot_string, "");
}

plt_line(arr, inc, years, x2, y2, scale, port)

double arr[], scale;
int inc, years, x2, y2;
int *port;

{
    int pc, i, Done;
    char temp[5];
    char plot_string[500];
    char c;

    Done = FALSE;
    while (!Done)
    {
        printf("\n\nSelect pen color:\n");
        printf("(W)ide black (N)arrow black (R)ed (G)reen (B)lue (M)agenta ");
        c = getch();
        switch(c)
        {
            case 'W' :
            case 'w' :
                pc = 1;
                Done = TRUE;
                break;
            case 'N' :
            case 'n' :
                pc = 2;
                Done = TRUE;
                break;
            case 'R' :
            case 'r' :
                pc = 3;
                Done = TRUE;
                break;
            case 'G' :
            case 'g' :
                pc = 4;
                Done = TRUE;
                break;
            case 'B' :
            case 'b' :
                pc = 5;
                Done = TRUE;
                break;
            case 'M' :
            case 'm' :
                pc = 6;
                Done = TRUE;
                break;
            default :
                printf("\aInvalid pen choice. Try again.\n");
                break;
        }
    }
    sprintf(plot_string, "SP%d;PA%d,%7.3f;PD", pc, inc, (arr[1]*scale));
    plot_str(plot_string, port);
    inc++;
    for (i=2; i<=years; i++)
    {
        sprintf(plot_string, "PA%d,%7.3f", inc, (arr[i]*scale));
        plot_str(plot_string, port);
    }
}

```

Source-code listing for program PLOT, continued.

```

        inc++;
    }
    sprintf(plot_string, "PU;SP0;PA%d,%d", x2, y2);
    plot_str(plot_string, port);
}

plot_stp(x2, y2, port)

int x2, y2;
int *port;

{
    char plot_string[50];

    sprintf(plot_string, "PA%d,%d;IW;", x2, y2);
    plot_str(plot_string, port);
}

/*****
/* File name      : READ_FLS.C
*****/

#include <stdio.h>
#include <io.h>
#include <graph.h>

#define TRUE 1
#define FALSE 0

read_crm(id, r, a, m, u, l, ia, in, years, first_yr, last_yr,
        q2, title, all_ok, ms)

FILE *id;
double *r, *a, *m, *u, *l, *ia, *in;
int *years, *first_yr, *last_yr;
double *q2, *ms;
char title[];
int *all_ok;
{
    char temp_str[10], line[81], iname[41];
    char *fgets();
    int i, strcmp();
    char *cf;
    double atof(), ddummy;

    *q2 = 0.0;
    _clearscreen;
    cf = fgets(line, 80, id);
    for (i=0; i<=80; i++)
    {
        if (line[i] != '\n')
            title[i] = line[i];
        else
        {
            title[i] = '\0';
            break;
        }
    }
    temp_str[0] = '\0';
    while (((i=strcmp(temp_str, "NUMB")) != 0) && !feof(id))
    {
        cf = fgets(line, 80, id);
        substr(temp_str, line, 1, 4);
    }
    if (!feof(id))
    {
        substr(temp_str, line, 24, 7);
    }
}

```

Source-code listing for program PLOT, continued.

```

    *ms = atof(temp_str);
    temp_str[0] = '\0';
    fill_sarray(id, r, &ddumy, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, a, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, m, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, u, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, l, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, ia, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, in, q2, first_yr, last_yr, years);
    printf("\a");
    return;
}
else
    *all_ok = FALSE;
return;
}

read_ind(id, a, m, u, l, in, years, first_yr, last_yr, q2, title,
        ms, cyr, cval, found, idrw)

FILE *id;
double *a, *m, *u, *l, *in;
int *years, *first_yr, *last_yr;
double *q2, *ms, *cval;
char title[], idrw[];
int *cyr, *found;

{
    char temp_str[10], line[81], iname[41];
    char *fgets();
    int i, slen, strcmp();
    char *cf, c;
    double atof();
    *q2 = 0.0;
    *found = FALSE;
    _clearscreen;
    _outtext("\nEnter core id number: ");
    gets(idrw);
    slen = strlen(idrw);
    if (slen < 2)
    {
        *found = FALSE;
        return;
    }
    cf = fgets(line, 80, id);
    for (i=0; i<=80; i++)
    {
        if (line[i] != '\n')
            title[i] = line[i];
        else
        {
            title[i] = '\0';
            break;
        }
    }
    temp_str[0] = '\0';
    while (((i=strcmp(temp_str, idrw)) != 0) && (!feof(id)))
    {
        cf = fgets(line, 80, id);
        substr(temp_str, line, 14, slen);
    }
    if (!feof(id))
    {

```

Source-code listing for program PLOT, continued.

```

    *found = TRUE;
    substr(temp_str, line, 27, 7);
    *ms = atof(temp_str);
    substr(temp_str, line, 36, 4);
    *cyr = atoi(temp_str);
    substr(temp_str, line, 42, 6);
    *cval = atof(temp_str);
    fill_sarray(id, a, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, m, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, u, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, l, q2, first_yr, last_yr, years);
    cf = fgets(line, 80, id);
    fill_sarray(id, in, q2, first_yr, last_yr, years);
    printf("\a");
    return;
}
)

read_tek(id, a, years, first_yr, last_yr, q2, title)

FILE *id;
double *a;
int *years, *first_yr, *last_yr;
double *q2;
char title[];

{
    char temp_str[10], line[81];
    char temp_line[80], c;
    char *fgets();
    int i;
    char *cf;
    extern int index();
    *q2 = 0.0;
    _clearscreen;
    cf = fgets(line, 80, id);
    substr(temp_str, line, 1, 4);
    *years = atoi(temp_str);
    substr(temp_str, line, 56, 4);
    *first_yr = atoi(temp_str);
    cf = fgets(line, 80, id);
    substr(temp_str, line, 2, 4);
    *last_yr = atoi(temp_str);
    cf = fgets(line, 80, id);
    fill_tarray(id, a, q2, first_yr);
    cf = fgets(line, 80, id);
    cf = fgets(line, 80, id);
    substr(temp_line, line, 19, 80);
    i = index(temp_line, " ");
    substr(title, temp_line, 1, i);
    printf("\a");
    return;
}

read_oth(id, a, years, first_yr, last_yr, q2, title, all_ok)

FILE *id;
double *a;
int *years, *first_yr, *last_yr;
double *q2;
char title[];
int *all_ok;

{
    char temp_str[10], line[81];
    char *fgets();

```

Source-code listing for program PLOT, continued.

```

int i, strcmp();
char *cf;
double atof();

*q2 = 0.0;
clearscreen;
cf = fgets(title, 80, id);
i = strlen(title);
title[i] = '\0';
cf = fgets(line, 80, id);
if (!feof(id))
{
    fill_sarray(id, a, q2, first_yr, last_yr, years);
    printf("\a");
    return;
}
else
    *all_ok = FALSE;
return;
}

/*****
/* File name      : S_PLOT.C
*****/

#include <stdio.h>
#include <graph.h>

#define TRUE 1
#define FALSE 0

g_line(j, xadd, yadd, xscale, yscale, years, arr, vmode, mult)

int j, years, vmode, mult;
float xadd, yadd, xscale, yscale;
double arr[];
{
    int i;
    char c;

    if (vmode > _TEXTMONO)
    {
        _outtext ("\nSelect color (B)lue (G)reen (C)yan (R)ed (M)agenta w(H)ite\n");
        _outtext ("          (P)ink (Y)ellow br(W)n gr(A)y lt.b(L)ue lt.gree(N)");
        c = getch();
    }
    if (vmode > _TEXTMONO)
    switch (c)
    {
        case 'B' :
        case 'b' :
            _setcolor(1);
            break;
        case 'G' :
        case 'g' :
            _setcolor(2);
            break;
        case 'C' :
        case 'c' :
            _setcolor(3);
            break;
        case 'R' :
        case 'r' :
            _setcolor(4);
            break;
        case 'M' :

```

Source-code listing for program PLOT, continued.

```

        case 'm' :
            _setcolor(5);
            break;
        case 'W' :
        case 'w' :
            _setcolor(6);
            break;
        case 'H' :
        case 'h' :
            _setcolor(7);
            break;
        case 'A' :
        case 'a' :
            _setcolor(8);
            break;
        case 'L' :
        case 'l' :
            _setcolor(9);
            break;
        case 'W' :
        case 'n' :
            _setcolor(10);
            break;
        case 'p' :
        case 'p' :
            _setcolor(12);
            break;
        case 'Y' :
        case 'y' :
            _setcolor(14);
            break;
        default :
            break;
    }
    moveto ((int)((xadd)*xsca'e), (int)((yadd-(arr[j]*mult))*yscale));
    for (i=j+1; i<=years; i++)
        _lineto((int)((xadd+i-1)*xscale),
                (int)((yadd-(arr[i]*mult))*yscale));
    _setcolor(8);
}
g_xyadd (xadd, yadd, first_yr, x1, y2, j)

float *xadd, *yadd;
int first_yr, x1, y2, *j;

{
    *xadd = first_yr-x1;
    if (first_yr > x1)
        *j=1;
    else
        *j= x1-first_yr;
    *yadd = y2;
    _outtext ("\\n");
}

g_labels(title, x_lab, y_lab, x1, y2, t1, t2, inc, choice)

char title[], x_lab[], y_lab[];
int x1, y2, t1, t2, inc, choice;

{
    int i, j, k;
    char indx[11];

    strcpy(indx, "NORMALIZED");
    _clearscreen(0);
    _settextposition(2, 46-(strlen(title)/2));
    printf("%s", title);
    _settextposition(21, 44-(strlen(x_lab)/2));

```

Source-code listing for program PLOT, continued.

```

printf("%s",x_lab);
k = strlen(y_lab);
if (k > 18)
    k = 18;
j = 0;
for (i=11-(k/2); i<=(11-(k/2))+k; i++)
    (
        _settextposition(i,3);
        printf("%c",y_lab[j]);
        j++;
    )
j = y2;
if(choice == 0)
    (
        for (i=4; i<=18; i+=inc)
            (
                _settextposition(i, 5);
                printf("%3d",j);
                j = j-t2;
            )
    )
else
    (
        for (i=6; i<=16; i++)
            (
                _settextposition(i,2);
                printf("%c",indx[i-6]);
            )
        _settextposition(4,5);
        printf(" 3");
        _settextposition(6,5);
        printf(" 2");
        _settextposition(9,5);
        printf(" 1");
        _settextposition(11,5);
        printf(" 0");
        _settextposition(13,5);
        printf(" -1");
        _settextposition(16,5);
        printf(" -2");
        _settextposition(18,5);
        printf(" -3");
    )

j = x1;
for (i=7; i<=80; i+=10)
    (
        _settextposition(19, i);
        printf("%4d",j);
        j = j+t1;
    )
)

g_ticks(xs,ys, j, k, l, m, w1,w2,w3,w4, xscale,yscale, x1, x2, y1, y2, t1, t2)

float xs, ys, j, l, xscale, yscale;
int k, m;
int w1, w2, w3, w4;
int x1, x2, y1, y2, t1, t2;

(
    int i, p;

    _rectangle(2, (int)(32*xs), (int)(28*ys),
                (int)(312*xs), (int)(140*ys));
    for (i=(int)(40*xs); i<(int)(312*xs); i+=(int)(8*xs))
        (
            _moveto(i, (int)(28*ys));
            _lineto(i, (int)(30*ys));
        )
    )

```

Source-code listing for program PLOT, continued.

```

        _moveto(i, (int)(139.5*ys));
        _lineto(i, (int)(138*ys));
    }
    for (i=(int)(72*xs); i<(int)(312*xs); i+=(int)(40*xs))
    {
        _moveto(i, (int)(28*ys));
        _lineto(i, (int)(32*ys));
        _moveto(i, (int)(139.5*ys));
        _lineto(i, (int)(136*ys));
    }
    for (i = 1; i<=k; i++)
    {
        p = (int)((28+((i-1)*j))*ys);
        _moveto((int)(32*xs), p);
        _lineto((int)(34*xs), p);
        _moveto((int)(312*xs), p);
        _lineto((int)(310*xs), p);
    }
    for (i=1; i<=m; i++)
    {
        p = (int)((28+((i-1)*l))*ys);
        _moveto((int)(32*xs), p);
        _lineto((int)(36*xs), p);
        _moveto((int)(312*xs), p);
        _lineto((int)(308*xs), p);
    }
    _settextwindow(22, 1, 23, 80);
    _setviewport((int)(32*xs), (int)(28*ys), (int)(312*xs), (int)(140*ys));
    _setcliprgn((int)(32*xs)+(int)((w1-x1)*xscale),
                (int)(28*ys)+(int)((y2-w4)*yscale),
                (int)(312*xs)-(int)((x2-w2)*xscale),
                (int)(140*ys)-(int)((w3-y1)*yscale));
}

/*****
/* File name      : WRITEF.C
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

writef(fp, prtarr, j, fyr1, ldec1)
int j, fyr1, ldec1;
FILE *fp;
double prtarr[];

{
    int i, j, jb, je;
    int done = FALSE;
    prtarr[j] = 99.99;
    jb = 1;
    je = (10 - (fyr1 % 10));
    while (!done)
    {
        if (i == j)
        {
            done = TRUE;
            break;
        }
        fprintf(fp, "%4d", fyr1);
        for (i=jb; i <= je; i++)
        {
            if (i == j)
            {
                done = TRUE;
                break;
            }

```

Source-code listing for program PLOT, continued.

```
        fprintf(fp, "%7.3f", prterr[i]);  
    }  
    fprintf(fp, "\n");  
    if (i <= 10)  
        fyr1 = fyr1 + i - 1;  
    else  
        fyr1 = fyr1 + 10;  
    jb = je + 1;  
    je = je + 10;  
}
```

Source-code listing for program MAKERAD.

```

/*****
/* Program name      : MAKERAD.C
/* Purpose           : This subroutine creates a radius file from a
/*                   : standard ring width file.
/* Input             : Standard ring width file in MM./100
/* Output            : Standard radius file used with COFECHA and
/*                   : other WRD TRL programs.
/* Programmer        : Michael L. Field
/* Created for       : Tree-Ring Laboratory
/*                   : Mail Stop #461
/*                   : U.S. Geological Survey
/*                   : Reston, Virginia 22092
/* Files used        : COMMON.C
/* Date Created      : 6/1/87
/* Modifications
/* Purpose           :
/* Date              :
/* Programmer        :
*****/

#include <stdio.h>
#define TRUE 1
#define FALSE 0

main()

(
FILE *id, *od;
FILE *setupinf(), *setupout();
char iname[41], oname[41], idnumber[9], line[101];
char c = ' ';
char tempstr[9], *cf;
char version[22];
char *fgets();
int atoi();

int i, j, value, years, elw;
int ci, first_year, last_year, crit_year;
int done = FALSE, first = TRUE;
unsigned int total;

crit_year = 0;
years = 0;
elw = 0;

cls();
printf("\n\n\n\n\n");
printf("
CREATE RADIUS FILE FROM RING-WIDTH FILE\n");
printf("
TREE-RING LABORATORY\n");
printf("
U.S. GEOLOGICAL SURVEY\n");
printf("
WATER RESOURCES DIVISION\n");
printf("
RESTON, VIRGINIA\n\n\n");
printf("
(Press enter to continue)");
gets(tempstr);
cls();
printf("\n\nRing width file name: ");
gets( iname );
if ((id=setupinf(iname, "r")) == (FILE *)NULL) /* set up input file */
(
printf("Program is terminating.\n\n");
exit();
)
printf("Does the file contain early and late ringwidths (Y/N, default=N)? ");
gets(tempstr);
if (tempstr[0] == 'Y' || tempstr[0] == 'y')
elw = 1;
printf("Radius file name: ");
gets( oname );

```

Source-code listing for program MAKERAD, continued.

```

if ((od=setupout(oname, "w")) == (FILE *)NULL) /* set up output file */
{
    printf("Program is terminating.\n\n");
    exit();
}
fgets(line, 100, id); /* read and discard title */
line[strlen(line)-1] = '\0';
fprintf(od, "%s\n", line);
while (!done)
{
    cf = fgets(line, 100, id);
    line[80] = '\0';
    if (feof(id))
    {
        fprintf(od, "%s %6u %4d %4d %4d %4d\n",
                idnumber, total, years, first_year, last_year, crit_year);
        done = TRUE;
        break;
    }
    substr(tempstr, line, 1, 8);
    if (strcmp(tempstr, idnumber) != 0)
    {
        if (!first)
        {
            fprintf(od, "%s %6u %4d %4d %4d %4d\n",
                    idnumber, total, years, first_year, last_year, crit_year);
        }
        substr(idnumber, line, 1, 8);
        substr(tempstr, line, 9, 4);
        first_year = atoi(tempstr);
        last_year = first_year - 1;
        total = 0;
        crit_year = 0;
        years = 0;
        first = FALSE;
    }
    substr(tempstr, line, 12, 1);
    if (!elw)
        j = 10 - (atoi(tempstr) % 10);
    else
        j = (5 - (atoi(tempstr) % 5)) * 2;
    for (i = 1; i < j + 1; i++)
    {
        substr(tempstr, line, (7 + (i * 6)), 6);
        if (strcmp(tempstr, "99900") == 0)
            break;
        total = total + atoi(tempstr);
        if (elw)
        {
            i++;
            substr(tempstr, line, (7 + (i * 6)), 6);
            total = total + atoi(tempstr);
        }
        last_year = last_year + 1;
        years = years + 1;
        if (crit_year == 0 && total >= 10000)
            crit_year = last_year;
    }
} /* end while */
fclose(od);
fclose(id);
} /* end */

cls()
{
    printf("\033[2J");
    fflush(stdout);
}

```

Source-code listing for program TRING.

```

/*****
/* Program name      : TRING.C
/* Purpose           : Creates a standard ring-width file.
/* Input             : From keyboard or measurement machine.
/* Output            : Standard ring-width file.
/* Programmer        : Michael L. Field
/* Created for       : Tree-Ring Laboratory
/*                   : Mail Stop #461
/*                   : U.S. Geological Survey
/*                   : Reston, Virginia 22092
/* Files used        : COMMON.C
/* Date Created      :
/* Modifications     :
/* Purpose           :
/* Date             :
/* Programmer        :
*****/

/* COMMAND DEFINITIONS
#define BOM -100 /* Beginning of Measurement Command */
#define EOM -200 /* End of measurement command */
#define BV -300 /* Command to back up one measurement */
#define FV -400 /* Command to go forward one measurements */
#define TRUE 1
#define FALSE 0

#include "stdio.h"
#include <bios.h>
#include <dos.h>
#include <conio.h>

main()
(
int finished, value, nvalues, maxvalues, byr, lyear;
int yrcnt, len_cnum, first, i, elw, fdone = FALSE;
int getvalue(), fseek(), value2, dec, mach;
int vdone = FALSE, apndr, apnde;
long fdec, fdecl;

char oname[41], rname[41], answer[2], cnum[9], temp_str[5], temp_str2[5];
char title[81], line[81], temp_name[41], elname[41];
char *fgets();

FILE *od, *td, *rd, *eld;
FILE *setupout();

/* Set line 0 for 9600, E, 1SB, 7BITS */
cls();
printf("\n\n\nWill this be (M)achine input or (K)eyboard? ");
gets(temp_str);
if (temp_str[0] == 'M' || temp_str[0] == 'm')
    mach = 1;
else
    mach = 0;

if (mach)
    _bios_serialcom(_COM_INIT, 0, (_COM_CHR7 | _COM_STOP1 | _COM_EVENPARITY
    | _COM_9600));
while (!fdone)
(
nvalues = 0;
byr = 0;
vdone = FALSE;
printf("Will this file contain separate early and late wood\n");
printf("measurements (Y/N)? ");
gets(answer);
if (answer[0] == 'Y' || answer[0] == 'y')
(
elw = TRUE;

```

Source-code listing for program TRING, continued.

```

    dec = 5;
    fdec = 7;
    fdecl = 9;
    lyear = FALSE;
}
else
{
    elw = FALSE;
    dec = 10;
    fdec = 11;
}
apndr = FALSE;
apnde = FALSE;
printf("Enter name of file to hold measurements(no .ext):");
gets(temp_name);
strcpy(online, temp_name);
strcat(online, ".rw");
strcpy(elname, temp_name);
strcat(elname, ".elw");
/* Setup file to receive values and return a file descriptor ( od ) */
if ( ( od = setupout(online, "w") ) == (FILE *)NULL )
{
    printf("Do you want to append to this file? (Y/N) ");
    gets(answer);
    if (answer[0] == 'N' || answer[0] == 'n')
    {
        apndr = FALSE;
        break;
    }
    else
    {
        od = fopen(online, "a");
        apndr = TRUE;
    }
}
if(elw)
{
    if ( ( eld = setupout(elname, "w") ) == (FILE *)NULL )
    {
        printf("Do you want to append to this file? (Y/N) ");
        gets(answer);
        if (answer[0] == 'N' || answer[0] == 'n')
        {
            apnde = FALSE;
            break;
        }
        else
        {
            eld = fopen(elname, "a");
            apnde = TRUE;
        }
    }
}
printf("Enter file description information (up to 80 characters)\n:");
gets( title );
if(!apndr)
    fprintf(od, "%s\n",title);
fclose(od);
if (elw)
{
    if (!apnde)
        fprintf(eld, "%s\n",title);
    fclose(eld);
}
finished = FALSE;
while ( !finished )
{
    printf("Enter control number: ");

```

Source-code listing for program TRING, continued.

```

gets( cnum );
len_cnum = strlen(cnum);
if( len_cnum < 8 )
{
    for( i=len_cnum; i<8; i++)
        cnum[i] = ' ';
    cnum[8] = '\0';
}
printf("Enter starting year: ");
gets(temp_str);
byr = atoi(temp_str);
yrcnt = byr;
td = fopen("tempfile.tmp","w");
printf("\nSend a -100 from the device's keypad to start recording.\n");
if (mach)
    while ((value = getvalue()) != BOM );
else
    while ((value = gval()) != BOM);
printf ("Starting to record measurments -- send a -200 from the keypad to end recording session.\n");
nvalues = 0;
maxvalues = 0;
printf("Year-> %4d",yrcnt);
if (elw)
    printf("\n Early value-> ");
else
    printf(" Value-> ");
while (!vdone)
{
    if (mach)
        value = getvalue();
    else
        value = gval();
    if (value == EOM)
    {
        vdone = TRUE;
        break;
    }
    switch( value )
    {
        case BOM: /* Another BOM command -- reposition to beginning*/
            printf("\nBacking up to first year.\n");
            nvalues = 0;
            maxvalues = 0;
            fseek( td, 0L, 0);
            yrcnt = byr;
            if (elw)
            {
                printf("Year-> %4d\n Early value-> ",yrcnt);
                lyear = FALSE;
            }
            else
                printf("Year-> %4d Value-> ",yrcnt);
            break;
        case BV: /* Back up to previous value */
            if( --nvalues < 0 )
            {
                nvalues = 0;
                maxvalues = 0;
                fseek( td, 0L, 0);
                yrcnt = byr;
                if (elw)
                {
                    printf("Year-> %4d\n Early value-> ",yrcnt);
                    lyear = FALSE;
                }
                else
                    printf("Year-> %4d Value-> ",yrcnt);
            }
    }
}

```

Source-code listing for program TRING, continued.

```

else
{
    if(elw)
    {
        if(lyear)
        {
            lyear = FALSE;
            fseek( td, -fdecl, 1 );
        }
        else
        {
            lyear = TRUE;
            yrcnt--;
            nvalues--;
            maxvalues--;
            fseek( td, -fdec, 1 );
        }
    }
    else
    {
        nvalues--;
        yrcnt--;
        maxvalues--;
        fseek( td, -fdec, 1 );
    }
}

printf("Backing up to year %d.\n",yrcnt);
if(elw)
{
    if(lyear)
    {
        printf("Year-> %d\n Late value-> ",yrcnt);
    }
    else
    {
        printf("Year-> %d\n Early value-> ",yrcnt);
    }
}
else
    printf("Year-> %d Value-> ",yrcnt);

break;
default: /* This is a value, store it */
if (!w)
{
    if (!lyear)
    {
        fprintf(td, "%d %d",yrcnt,value);
        if (mach)
            printf("%d Late value-> ",value);
        else
            printf(" Late value-> ");
        lyear = TRUE;
    }
    else
    {
        fprintf(td, " %d\n",value);
        yrcnt++;
        if (mach)
            printf("%d\nYear-> %d\n Early value-> ",value,yrcnt);
        else
            printf("Year-> %d\n Early value-> ",yrcnt);
        nvalues++;
        lyear = FALSE;
    }
}
}

```

Source-code listing for program TRING, continued.

```

else
(
    fprintf( td, "%4d %4d\n", yrcnt, value);
    nvalues++;
    yrcnt++;
    if (mach)
        printf("%5d\nYear-> %4d Value-> ", value, yrcnt);
    else
        printf("Year-> %4d Value-> ", yrcnt);
    )
if( nvalues > maxvalues )
    maxvalues = nvalues;
)
)
while( nvalues++ < maxvalues )
    fprintf( td, "9999 999\n");
value2 = 0;
printf("End of recording session\n");
fprintf( td, "9999 999\n");
fclose( td );
first = 1;
td = fopen("empfile.tmp", "r");
od = fopen(online, "a");
if (elw)
    eld = fopen(eldname, "a");
while(value != 999)
(
    fgets(line, 80, td);
    substr(temp_str, line, 6, 4);
    value = atoi(temp_str);
    if (value == 999)
        break;
    substr(temp_str, line, 1, 4);
    yrcnt = atoi(temp_str);
    if (elw)
    {
        substr(temp_str, line, 11, 4);
        value2 = atoi(temp_str);
        if (value2 == 999)
            break;
        if ((yrcnt % 5) == 0 || first == 1)
        {
            if (!first)
                fprintf(eld, "\n");
            fprintf(eld, "%s%4d", cnum, yrcnt);
        }
        fprintf(eld, "%6d%6d", value, value2);
    }
    if ((yrcnt % 10) == 0 || first == 1)
    {
        if (!first)
            fprintf(od, "\n");
        fprintf(od, "%s%4d", cnum, yrcnt);
        first = 0;
    }
    fprintf(od, "%6d", value+value2);
)
if (elw)
(
    if (yrcnt % 5 < 4)
        fprintf(eld, " 99900\n");
    else
        fprintf(eld, "\n");
)
if(yrcnt % 10 < 9)
    fprintf(od, " 99900\n");
else
    fprintf(od, "\n");

```

Source-code listing for program TRING, continued.

```
        fclose(td);
        fclose(od);
        if(elw)
            fclose(eld);
        printf("Do you want to record more measurements? (Y/N): ");
        gets(answer);
        if ( answer[0] == 'N' || answer[0] == 'n' )
            finished = TRUE;
        else
        {
            vdone = FALSE;
            finished = FALSE;
        }
    }
    printf("Do you want to make another file (Y/N)? ");
    gets(answer);
    if ( answer[0] == 'N' || answer[0] == 'n' )
        fdone = TRUE;
    }
    unlink ("tempfile.tmp");
}

gval()
{
    char temp[5];
    int value;

    gets(temp);
    value = atoi(temp);
    return value;
}

getvalue()
{
    unsigned far *com_base;
    char buffer[4];
    int i, value, sign;

    com_base = (unsigned far *) 0x00000400L;
    i = 0;
    while ( i < 4)
    {
        if ((unsigned char) inp(*com_base+5) & 0x01)
            buffer[i++] = (unsigned char) inp(*com_base);
    }
    sign = 1;
    if (buffer[0] == ' ')
    {
        sign = -1;
        value = 0;
    }
    else
        value = buffer[0] - '0';
    for (i=1; i<4; i++)
        value = (10*value) + (buffer[i] - '0');
    value = sign * value;
    return (value);
}

cls()
{
    printf("\033[2J");
    fflush(stdout);
}
```

Source-code listing for files common to all programs.

```

/*****
/* substr.c -- substr
/*
/* 'substr' extracts a substring from a string given a starting
/* position and length.
*****/
substr(substring, string, start, length)
char *substring;
char *string;
int start;
int length;
{
    int i;
    start--;
    string += start;
    for (i=0; i < length && *string != '\n' && *string != '\0'; i++)
        *substring++ = *string++;
    *substring = '\0';
}

#include "stdio.h"

#define TRUE 1
#define FALSE 0

/*****
/* setupinf.c setupinf(6file name)
/* sets up a file for reading
/* returns file number -- NULL if problems
*****/
FILE *setupinf ( fname, mode )
char *fname;
char *mode;
{
    FILE *fd;

    char answer[5];
    char *gets();
    if ((fd = fopen (fname, mode)) != (FILE *)NULL )
        return (fd);
    else
    {
        fclose ( fd );
        printf("Unable to open file <fs> -- file probably does not exist, try again.");
        return((FILE *)NULL);
    }
}

/*****
/* setupout.c setupout(file name)
/* sets up a file for writing
/* returns file number -- NULL if problems
*****/
FILE *setupout ( fname, mode )
char *fname;
char *mode;
{
    FILE *fd;

    char answer[5];
    char *gets();
    if ((fd = fopen (fname, "r" )) == (FILE *)NULL )
    {
        fclose( fd );
    }
}

```

Source-code listing for files common to all programs, continued.

```
        fd = fopen ( fname, mode );
    }
else
{
    fclose ( fd );
    printf("File <%.s> already exists.\n", fname );
    printf("Do you want to overwrite it? (Y/N, default=Y) ");
    gets (answer );
    if ( answer[0] != 'N' && answer[0] != 'n' )
    {
        if (( fd = fopen(fname, mode)) == (FILE *)NULL )
        {
            printf("Unable to open file <%.s>.\n",fname);
            return((FILE *)NULL);
        }
    }
    else
        return((FILE *)NULL);
}
return(fd);
}
```

